NPL
National Physical Laboratory

NPL REPORT MS 36

GOOD PRACTICE GUIDE ON INDUSTRIAL SENSOR NETWORK METHODS FOR METROLOGICAL INFRASTRUCTURE IMPROVEMENT

LUO, Y.
HARRIS, P.
WRIGHT, L.
JAGAN, K.
KOK, G.
COQUELIN, L.
ZAOUALI, J.
EICHSTÄDT, S.
DORST, T.
TACHTATZIS, C.
ANDONOVIC, I.
GOURLAY, G.
YONG, B.

OCTOBER 2021

GOOD PRACTICE GUIDE ON INDUSTRIAL SENSOR NETWORK METHODS FOR METROLOGICAL INFRASTRUCTURE IMPROVEMENT

Yuhui Luo, Peter Harris, Liam Wright and Kavya Jagan
Data Science Department, NPL, UK

Gertjan Kok
VSL, The Netherlands

Loic Coquelin and Jabran Zaouali
LNE, Frace

Sascha Eichstädt
PTB, Germany

Tanja Dorst
ZeMA, Germany

Christos Tachtatzis, Ivan Andonovic and Gordon Gourlay
University of Strathclyde, UK

Bang Xiang Yong
University of Cambridge, UK

ABSTRACT

This guide presents some specific methods to identify the measurement coverage and accuracy required for process output quality targets in industrial sensor networks. It also describes some other methods of metrological data processing for industrial process optimization, focusing on aspects of redundancy, synchronization and feature selection applied to data affected by measurement uncertainty. A testbed, concerning a radial forge at the University of Strathclyde's Advanced Forming Research Centre in the UK, is used as an illustration for this guide.

This guide is a deliverable of the project 17IND12 Met4FoF "Metrology for the Factory of the Future" (http://www.met4fof.eu) funded by the European Metrology Programme for Innovation and Research (EMPIR).

ISSN 1754-2960

https://doi.org/10.47120/npl.MS36

National Physical Laboratory
Hampton Road, Teddington, Middlesex, TW11 0LW

Approved on behalf of NPLML by
Kevin Lees, Group Leader (Data Science Department).

CONTENTS

1 INTRODUCTION

1.1 BACKGROUND

Artificial intelligence, robotics and condition monitoring are a few of the key enablers for the so-called ‘Factory of the Future’, pushing quality control and assurance in the direction of greater automation and higher accuracy. Industrial sensor networks and the associated data processing pipelines are the infrastructure underlying these capabilities. Benefitting from the advancement of digital technologies, sensor networks have already been deployed on many manufacturing sites and have motivated associated research and development activities.

Compared with small-scale sensor networks, the processing of measured data is more challenging for a large sensor network, or for a structurally complicated network of sensors with unknown number of latent variables and possible high levels of interaction between the sensors. Industrial sensor networks, however, often have these properties. There are often over a hundred sensors in an industrial sensor network. Traditional mathematical modelling approaches, which require an understanding of the physical principles of the system, become impractical for such cases. Machine learning (ML) methods, also known as data-driven methods, provide a feasible alternative and have received considerable attention from practitioners in recent times. Instead of requiring explicit knowledge of the system, ML methods can be regarded as a black-box approach. The word “learning” refers to the capability of a ML method to learn from training data, which is typically achieved by optimizing an objective cost function. Depending on whether the training data includes labels, ML methods are categorized as supervised or unsupervised. In supervised learning, a ML model provides a mapping between input sensor data and the output target variable (considered as a label) by learning from the training data. In unsupervised learning, in comparison, the model provides the underlying structural properties of the input sensor data. There are advantages and disadvantages associated with the use of ML. An obvious advantage lies in the adaptive nature of ML methods that removes the need to explicitly model the physical system. A disadvantage can come from a lack of “interpretability” of the model and the difficulty in evaluating associated uncertainty information.

Uncertainty evaluation is an essential requirement in the processing of measured data, which underpins considerations of metrological traceability and quality-assurance. In metrology (or measurement science), the “Guide to the expression of uncertainty in measurement” (GUM) [GUM, 2008] provides a standard framework for evaluating measurement uncertainty that is both model-based and probabilistic. The focus of the GUM is on uncertainty propagation for a linear system or an approximately linearized system in the case that the system is mildly nonlinear. When a linear approximation is not acceptable, the GUM permits “other analytical or numerical methods” for uncertainty propagation, such as a Monte Carlo method. It is well-known that nonlinearity is one of the distinguishing features of ML models. Especially for those ML models based on neural networks with multiple layers and complicated structures, various forms of nonlinearity are embedded within the models, making the standard approach to uncertainty evaluation difficult.

Besides the challenge of uncertainty evaluation in ML, there are other practical factors to be considered in the data processing pipeline. For example, from a manufacturing perspective, the availability of an insufficient amount of training data, and quality concerns such as mis-labelling of training data and lack of metadata, are often encountered. In the context of sensor networks, the quantification of redundancy in the sensor network measurements, the presence of jitter effects in sensor outputs, and problems of synchronization errors for multiple sensors, are frequently raised concerns. From the algorithmic point of view, there are various ML methods available. The selection of an appropriate ML method, understanding the uncertainty of the ML model used, and the robustness of the ML algorithm, are all current topics of research

with practical applications. For practitioners, so far there is limited guidance on the data processing pipeline, which motivates the writing of this Good Practice Guide and using the testbed for illustration.

1.2 AIM OF THIS GUIDE

Limited guidance currently exists to address the challenges described above for ML methods used in the data processing pipeline for industrial sensor networks. In this guide, some elements of current good practice are provided about methods for processing metrological data in that context. The aim, particularly, is to address issues such as the identification of the measurement coverage and accuracy required for process output quality targets in industrial sensor networks. A testbed, referred to as the "STRATH testbed", in the form of a radial forge located at the University of Strathclyde's Advanced Forming Research Centre (AFRC) in Glasgow, UK, is used as an illustration for this guide. This guide is not a comprehensive user manual for how to use ML in the data processing pipelines for industrial sensor network. It provides example-assisted metrology guidance to show how typically ML may be applied, i.e., how things may be done properly.

1.3 STRATH TESTBED

The testbed considered in this guide is a radial forge, as shown in Figure 1. Radial forging has been widely used in areas such as automotive, aerospace, rail, and medical manufacturing. The forge is a four-hammer machine, the maximum forging force is 1,500 kN, the hammer speed is 1,200 stroke/min, and the part rotational speed is approximately 44 rpm with a programmable range of 40-70 rpm. The hammers are concave to match product specification. The testbed is equipped with around 100 sensors used to record the forging process with a sampling interval normally set to 10 ms. These sensors include those dedicated to process output and those dedicated to machine maintenance. The four hammers are aggregated into two sensor diagnostics (left and right) that record hammer force, position, and speed. There are also pyrometers to record temperature at various locations, e.g., in the forging box.

Figure 1: The radial forge housed at the Advanced Forming Research Centre (part of the University of Strathclyde, Glasgow, UK).

There are three datasets available from the testbed corresponding to one validation run (with no changes to machine settings) and two designs of experiment. In the first dataset, 81 parts were forged from two material batches of 50 parts each over one day of operation. It takes roughly 3 minutes to forge each part and so, with a sampling interval of 10 ms, a time series of roughly 18,000 samples is provided by each sensor and for each forged part or experiment. The second and third datasets contain experimental data for 50 and 46 parts, respectively, with three controlled variations, namely of billet temperature at the entry to the forge, chuck head axial feed speed and hammer radial feed speed. The objective of applying such controlled variations is to mimic a larger range of input conditions to the testbed.

After forging, each completed part was measured in a Coordinate Measuring Machine (CMM) to obtain dimensional data, which are compared with target specifications and tolerances. The CMM makes 16 dimensional measurements. Some of these measurements correspond to the same dimension but measured at different part orientations. Therefore, redundancy in the dimensional measurements is expected. A part before and after forging is shown in Figure 2. Besides the target part dimensions, another key target process output is energy consumption for the forging of each part, which contributes to manufacturing cost and demonstrates the use of sensor network data for process optimization.

Figure 2: Part before (preform) and after forging.

The STRATH testbed serves as a good practical example for measurement data processing pipelines for the following reasons. Firstly, the forging process is characterized by many sensors measuring different quantities. There is no explicit knowledge of the relationships between the sensor measurements and the part dimensions, making ML methods a clear choice for the prediction task. Secondly, metadata about the sensor usage is available, but there are no clear markers for the forging state, leading to the need to consider alignment and redundancy evaluation in the data processing pipeline. Finally, the training data provided by the testbed is modest in size, typical of manufacturing applications compared to the application of ML in idealized scenarios where large volumes of training data are often available.

2 LITERATURE ON MACHINE LEARNING METHODS FOR MANUFACTURING

2.1 WIDELY AVAILABLE GENERAL SURVEYS

There are open discussions on the application of ML methods in industrial sensor networks for the “Factory of the Future” on the internet and in publications, such as the review paper [Wuest, 2016]. Most of the literature is of a narrative type, emphasizing the industrial application and benefits. For ML in manufacturing, those benefits can be summarized in the following ways:

1. Improving operational efficiency and lowering costs
2. Reducing maintenance costs and improving reliability
3. Reducing inventory levels and waste
4. Improving quality control on the production line
5. Improving the design of new products

For example, it is reported in [Columbus, 2020] that, as the most up-to-date techniques:

- Nokia has introduced a video application that uses ML to alert an assembly operator if there are inconsistencies in the production process
- General Motors collaborated with Autodesk to implement generative design algorithms that rely on ML techniques to factor in design constraints and provide an optimized product design
- Thales SA, a leading supplier of electronic systems to a wide spectrum of industries, is using ML to predict preventative maintenance for high-speed rail lines throughout Europe
- The BMW Group uses ML to evaluate component images from its production line, allowing it to spot, in real-time, deviations from quality standards
- Nissan is piloting the use of artificial intelligence to design new models in real-time, hoping to reduce time-to-market for the next-generation model series
- Canon has invented an advanced Asset Defect Recognition system that brings new levels of quality control to its manufacturing centers

However, in general, ML for manufacturing is still a new technology under development, and these studies are limited in their depth and provide little metrological guidance.
There are also academic reviews on ML techniques for manufacturing, e.g., [Wuest, 2014] and [Namuduri, 2020]. For example, in [Namuduri, 2020], a review on the possibility of applying deep learning for predictive maintenance is provided. The relationship between the sensor type, the selection of deep learning algorithm and the application area in manufacturing is suggested but no example is described. Other publications discuss in more detail specific technologies in sensor networks for industrial processes. In [Shrouf, 2014], for example, the technology of the 'Internet-of-Things' (IoT) for the 'Smart Factory' is studied. A reference architecture for IoT-based smart factories is presented and an approach for energy management is proposed. There are two characteristic strands in all the ML methods for manufacturing: the availability of 'big data' from the sensor measurements and the ability of ML algorithms to learn from the data towards the objective function.

2.2 EXAMPLES OF ML METHODS FOR MANUFACTURING

There is limited open literature on the detailed application of ML for manufacturing due to concerns around the protection of intellectual property and trade secrets. In [Wuest, 2014], an approach to increase quality and improve process control in manufacturing is proposed. It utilizes a combination of cluster analysis and supervised ML. Interestingly, the example given is a hot forging process from raw material to the part after hardening, which resembles to a certain degree the testbed considered in this work. It is reported that the analysis of product and process state at various checkpoints offers the opportunity for process parameter adjustment, as illustrated in Figure 3. The state information can subsequently be used for machinery warning monitoring and product quality control purposes. However, the determination for each product state is associated with varied numbers of dependences. Not all dependences are known, and it is impossible to model the input-output relationships. ML methods thus apply to learn from past data. A Support Vector Machine (SVM), which is a type of supervised learning method, is suggested to classify the process/product state. However, there are neither details about the features used to train the SVM nor results presented in the paper. Issues such as data pre-processing and other metrological concerns are not addressed. Despite the lack of more details, this work shows a possible way for applying ML in

manufacturing, and it also points to concerns about data completeness and potential problems of sampling methods for the creation of the training data.

Another example of a ML method applied in the context of sensor networks is in semiconductor manufacturing. In this application, integrated circuit fabrication at the nanoscale is not entirely deterministic and tiny uncontrollable variations or defects in manufacturing lead to significant uncertainty in the behavior of the whole circuits, where a high level of quality control is always the top priority. In [Chen, 2017] and [Chen, 2019], ML methods are applied for modelling variations and yield improvement in the fabrication process. To predict the variation in fabrication, various optimization and test methods have been proposed. These tests are complicated, expensive, and time-consuming. Because of the spatial variation of wafers and dies, to obtain a parameter, it is necessary to design and fabricate numerous test structures. With ML methods, the costs of such tests can be significantly reduced.

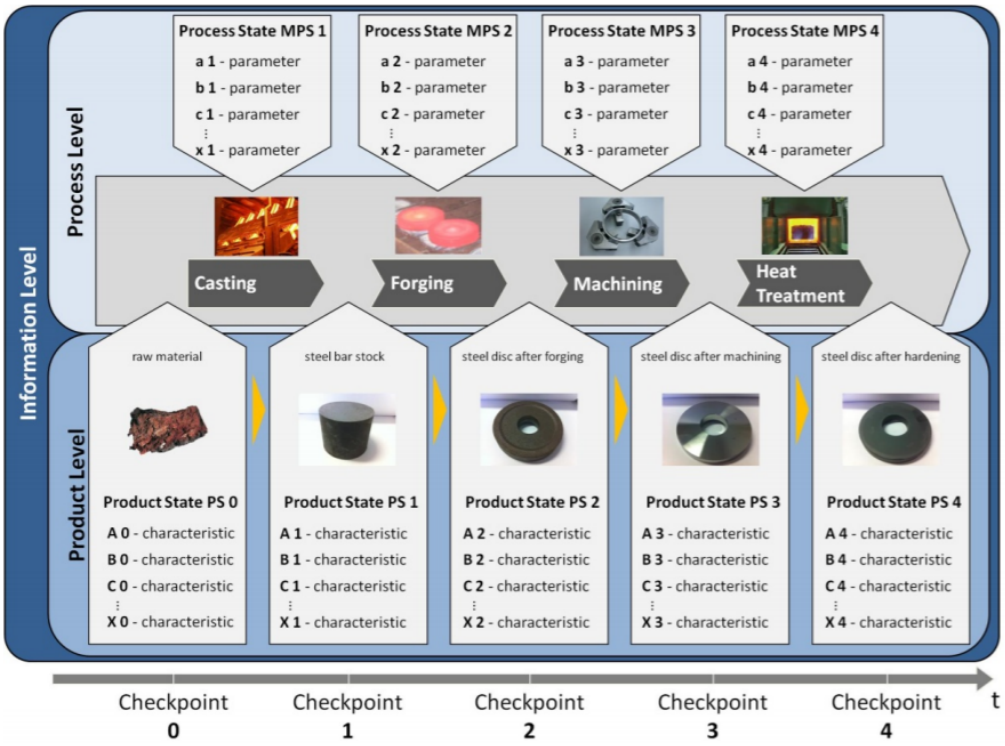

Figure 3: Process flow, taken from [Wuest, 2014].

The thesis of [Moller, 2017] aims to investigate the possibilities and limitations of using ML methods for industrial quality control. Some ML methods are briefly introduced, such as convolution neural networks and reinforcement learning. Some issues in ML for industrial applications are mentioned, e.g., tuning of parameters in neural networks. Three interviews with manufacturers are used as examples of possible applications with ML methods. However, the numerical illustration of audio classification is not linked to any practical testbed.

In [Carlos, 2018], ML methods in high conformance manufacturing environments are investigated with two automotive cases studies. The detection of defects is presented as a problem of binary classification. Several ML feature selection techniques are discussed, including regularization and filtering type feature selection. It provides a good algorithmic overview of feature selection algorithms although the metrological prospect in manufacturing is not mentioned.

In parallel to development of the STRATH testbed, the “ZeMA testbed”, located at the research institute of ZeMA, Germany, has been studied in the Met4FoF project. The ZeMA testbed is used for condition monitoring, lifetime prognosis and end-of-line quality control of electromechanical cylinders (EMCs). It serves as an alternative good example in demonstrating the pipeline of data processing from metrological prospect. Note that one of the characteristics of the ZeMA testbed is its high sampling rate of up to 1 MHz, leading to large volumes of data to be analysed. Interested reader can refer to the publications of [Schneider, 2018A], [Schneider, 2018B], [Helwig, 2017], [Dorst, 2021], and [Dorst, 2019] for more details.

To summarize, the literature on ML for industrial sensor networks often focuses on specific manufacturing processes, leading to variations in data processing pipelines. Elements including the level of user knowledge, size of available data and data quality represented by the associated uncertainty, are important factors in ML. Despite the differences in applications, it is desirable to have general guidance for metrological data processing in industrial sensor networks.

3 DATA PROCESSING PIPELINE

3.1 DATA PROCESSING PIPELINE IN GENERAL

There is no standard data processing pipeline for industrial sensor networks, as frameworks for the pipeline may vary depending on the nature of the manufacturing process. In this section, a possible data processing pipeline for industrial sensor networks is discussed, which can be considered as a summary for different possible applications, including the ZeMA testbed [Schneider, 2018A], [Schneider, 2018B], [Helwig, 2017], [Dorst, 2021], and [Dorst, 2019], and the STRATH testbed, which have been studied within the Met4FoF project. The pipeline involves the following steps:

Step 1: Establish development goal

The first step in a data-driven project for an industrial sensor network is to establish the development goal. Prediction or classification ability is often expected. Factors such as the availability and the size of a training dataset, data quality, the associated uncertainty, and the required accuracy of the output can influence the construction of the data processing pipeline. For example, in a prediction or regression problem, when the size of the training dataset is small, deep learning ML methods may not be directly applicable because they typically require a large amount of training data. Practitioners may need to synthesize data to assist further training or opt for other solutions. In another example, for product quality monitoring, binary classification may not be sufficient for the accuracy required. A detailed understanding of the goal and the industrial environment is therefore necessary before the start of any operations.

Step 2: Data synchronization

The next step in the data processing pipeline is the synchronization of the raw data captured from industrial sensors, which is fundamental to further analysis. For some manufacturing applications, the sensors are controlled by a central control unit and the sensor data can be considered as perfectly synchronized (or synchronized to an acceptable degree). For other applications, however, the lack of adequate synchronization may need consideration. To give a few examples, when the timing or control signals are triggered from different units, or in a large-scale sensor network when delays from data communications vary, or when the time resolutions of sensor data differ, synchronization effects are unavoidable. Since concerns about synchronization are frequently raised in manufacturing IoT applications, a subsequent section with a metrological focus is dedicated to this topic.

Step 3: Data curation

After synchronization, depending on the data quality and the associated uncertainty, data cleaning could be the next step in the data processing pipeline. But, except for addressing obvious errors in numeric data or treating faulty data due to various reasons, the understanding of ‘cleaning’ can be subjective, and the corresponding operations to be applied to data can differ depending on the degree of ‘cleaning’ required. For example, in monitoring the noise level using an acoustic sensor, an outlier could be an error in recording due to instrumentation. But it is also possible that it represents a valid change in noise level. When the machinery is restarted from failure or is in a transient state, data quality often needs attention.

Step 4: Redundancy evaluation

Redundancy evaluation is another key step in the pipeline. This step is of particular importance to practitioners due to the reason that the data from industrial sensor networks is typically classified as ‘big data’ because of the data size and its complexity. Note that the step of redundancy evaluation is often used interchangeably with dimensional reduction in many publications. Their objectives are to retain useful information with a more compact data representation. However, although a wide range of publications have been dedicated to dimensional reduction, the meaning of ‘redundancy’ in an industrial sensor network is less clear in a metrological setting. Techniques for dimensional reduction, while providing a reduced dataset, can also remove information that is crucial for the development goal. For example, the classical dimensional reduction technique of Principal Component Analysis (PCA) extracts the directions where the major variance of data lies by retaining those eigenvectors corresponding to the largest eigenvalues. But it is possible that, in a prediction task, useful features are embedded in those directions which correspond to the eigenvectors associated with the smallest eigenvalues. Hence, ‘redundancy’ removal, especially in the early stages of the data processing pipeline, should be treated with extra consideration to prevent potentially useful information from being lost. In this guide, more detailed guidance about metrological redundancy evaluation is provided in section 4.

Step 5: Feature extraction

For a ML project, the stage of learning or classification from training data is a necessary step. The way of training or learning may take different forms. One way is to learn directly from the training data, analogously to black box modelling. While black box modelling is appealing, feeding data as the input to obtain results as the output, in an industrial sensor network learning directly from the output of manufacturing processes is a challenging task, as the output is in many cases a time series. Meanwhile, it is possible that there is a certain degree of understanding of the machinery and mechanism that can be utilized for prediction. Therefore, instead of feeding the time series directly into the regressor or classifier, feature extraction is a common step in the data processing pipeline.

In most applications, however, it is a challenging task to know the features needed for the development goal. Features in the time domain or in a transformed space, such as the frequency domain, are often extracted to the candidate pool. Consider a vibration sensor as an example. For the time frame of interest, the maximum value of the signal, root-mean-square value and their ratio are often used as features in a conformance maintenance task. The vibration signal can also be transformed into the frequency domain. The frequency range of the spectrum or the amplitudes at certain frequencies could be another set of features to put into the candidate pool. Therefore, feature extraction often results in many features, especially when there is no prior knowledge of the physical system. For an industrial sensor network in which big data is typically changing dramatically, the number of features extracted can easily grow from tens to thousands. Feature selection (below) then becomes necessary, as it reduces

computational complexity, improves robustness, and helps in understanding the physical system.

Step 6: Feature selection

It is well-known that the prediction ability of a regressor or classifier is determined by the amount of information embedded in the features. The objective of feature selection is to select 'good' features for the development goal. In general, a feature is 'good' if, firstly, it provides information to the development goal and secondly it is not redundant with respect to other good features. In other words, using the Pearson correlation coefficient as a goodness measure, a good feature is expected to be highly correlated with the class labels or target output while it is weakly correlated with other good features [Carlos, 2018]. There are many algorithms available for features selection, ranging from filter type algorithms, e.g., ReliefF in [Robnik-Sikonja, 2003], to decision tree type algorithms, e.g., random forests [Breiman, 2001] and Bayesian additive regression trees [Hill, 2020].

Step 7: Algorithm selection

In the development stage, it is possible to put several algorithms on trial and select the one that works best, which is the process phase of algorithm/model selection. There are a range of factors in algorithm selection such as size of training data, algorithm robustness, numerical stability, computational complexity, etc. The selection of a suitable algorithm depends on the application. For example, when real-time response is required, algorithmic computation complexity is of high priority. On the other hand, if high precision is desired, the prediction accuracy is the most important element to be considered. Some ML methods such as deep learning need a large amount of training data. While for a small dataset, Bayesian methods generally work fine, although the results can be sensitive to the choice of prior distribution. In this guide, the aim is not to compare the available feature selection algorithms. Instead, the issue of uncertainty in feature selection will be addressed in a later section from a metrological perspective.

Step 8: Cross validation

Following feature selection, model validation in general is the last step in the data processing pipeline. Methods for validation such as k-fold cross validations are applied in many publications such as [Dorst, 2021]. It is strongly suggested to use cross validation to avoid overfitting and estimate the success of the model on new data.
In the rest of this section, the STRATH testbed will be used as an example and a few of the steps that are commonly found in the data processing pipelines for many applications will be further studied in later sections. It is important to note that, with new developments in signal processing, novel technology in data storage and management, and advances in computer science, the processing steps are expected to alter and adapt as time goes by.

3.2 STRATH TESTBED

For the STRATH testbed, metal parts are forged sequentially. The goal of the ML analysis is to predict the part dimensions and to optimize the process based on the sensor data. There are a large number of sensors operating in the testbed, producing a dataset consisting of 99 individual time series. To reduce the amount of data to be considered, it is beneficial to divide the overall forging process into processing phases, because the sensors continuously record data from beginning to end in the process. Yet, useful information may only be present during a small portion of the overall time. For example, the force sensor in the forging box, which monitors the force applied during forging, provides limited information when a part is heated in the induction coil. The whole forging process can be divided into three phases: the phase of

heating using the induction coil, the phase of pick-up and transfer from the induction coil to the forging box, and the phase of forging into the predefined shape by hammers.

Although sensor signals in the STRATH testbed are fully synchronized, there are no labels or metadata information to indicate the different states in the whole forging process. Considering the process and the objective of part dimension prediction and process optimization, the data processing pipeline can be summarized in the flowchart in Figure 4.

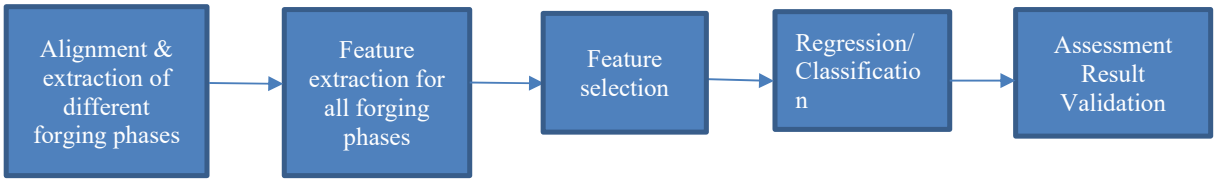

Figure 4: Flowchart of data processing pipeline for the STRATH testbed.

The first phase in the data process pipeline, namely alignment and extraction of different forging phases, can be regarded as synchronization in the wider sense. There may exist alternative ways to define the processing phases in the production. However, the segmentation of heating, pick-up and forging phases matches the sequence in the overall forging procedure and enables us to analyse data with better insight. To do so, sensor signals can be used to mark the start and end of each phase. For example, the on/off state of power for the induction coil can be used to signal the start and the end of the heating phase. For the forging phase, there are several sensor signals that can be applied for segmentation. One obvious choice is the sensor which records the force applied to the part. The active period of the forging phase can be detected with thresholding of that force sensor. Another choice is the position sensors which record the nominal and actual positions of the chuck head axes. There are also other selections such as the sensors which monitor the speed of the chuck head in certain axis. They can be used to keep track of the movement of the part during the forging. Another possible indicator is the two peaks of the actual value of the pyrometer which measures the temperature of the part in and out of the forging box. Note that all these indicators are valid indicators with logical support from knowledge of the forging process. For a typical part in the STRATH dataset, the sensor signals described above in the neighbourhood of the forging phase are shown in Figure 5.

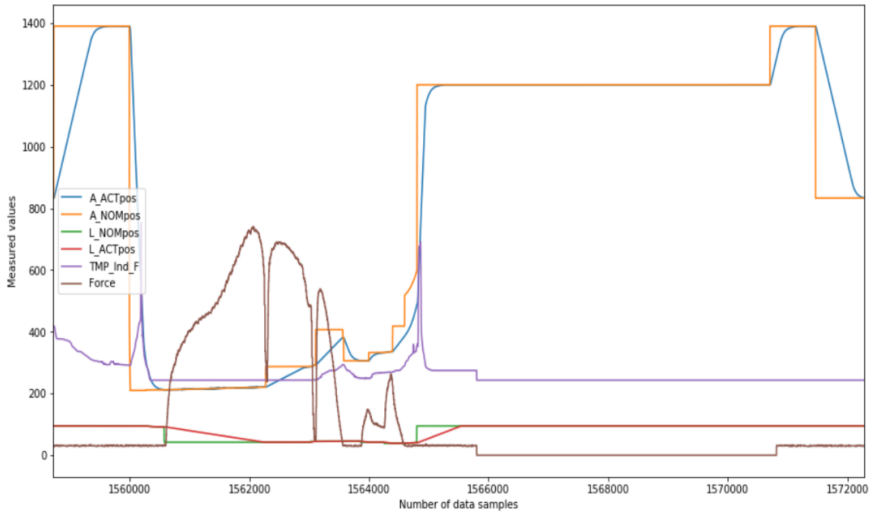

Figure 5. A few sensor signals in the neighbourhood of forging the phase.

However, all these sensors have different working mechanisms, leading to varying degrees of accuracy in the segmentation. In most cases, for example, the force applied is close to zero before forging starts. But it is not necessarily so in all cases. Segmentation of the forging phase therefore depends on the value of thresholding, which results in another type of uncertainty. The selection of a proper segmentation requires detailed understanding of the physical working mechanism of the testbed. This is a crucial point as the STRATH testbed demonstrated. That is, understanding of the system helps, even in a data-driven project.

In Figure 6, the forging durations obtained using the signals recorded by the position sensors and the pyrometers are compared. Although the patterns have a strong similarity, the differences may affect subsequent feature extraction, selection and the prediction accuracy.

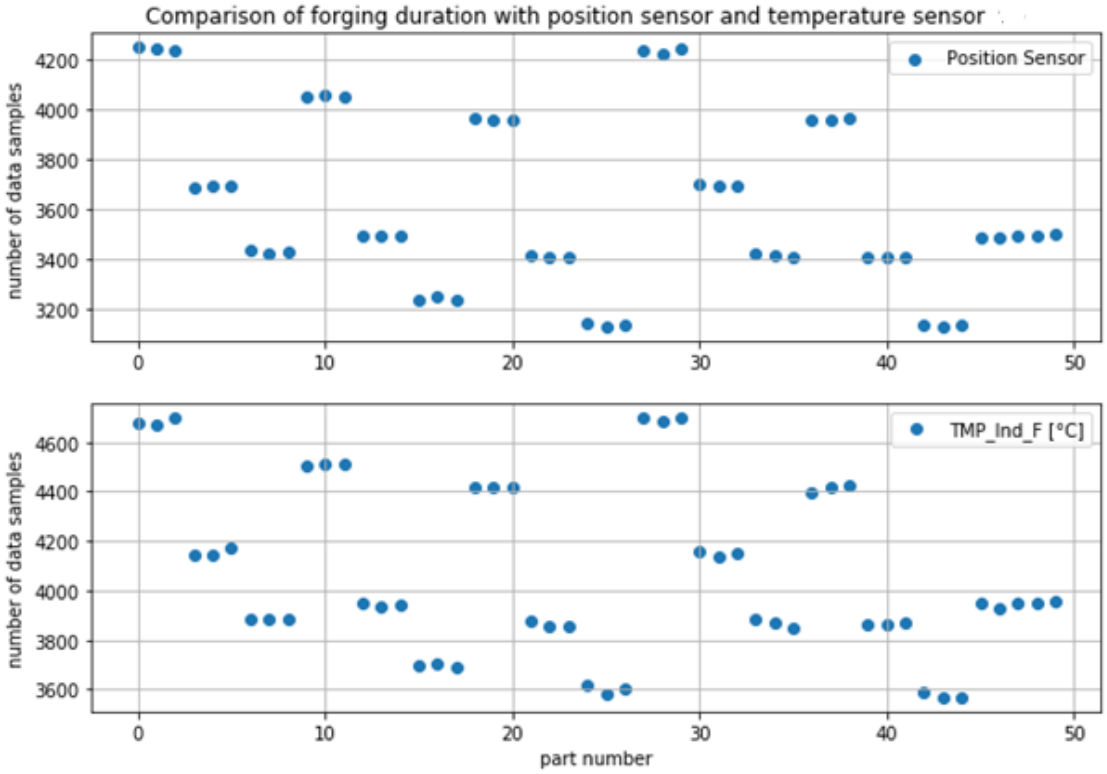

Figure 6. Forging duration in number of samples based on different segmentations.

After segmentation and using metadata information, the numbers of sensors considered in each phase are greatly reduced. In the heating phase, the number of sensors considered is 13 instead of the original 99 sensors. In the forging phase, the number of sensors to be considered is reduced to 62. For the pick-up phase, only a handful of sensors needs further analysis. Although this is not redundancy removal in the strict sense, the amount of data to be further analysed is nevertheless greatly reduced by the segmentation.

For feature extraction, there are some features drawn from the statistics of the signals and others based on an understanding of the testbed. In the case of a lack of knowledge of what types of features are needed for the regression or classification goal, statistical features are candidates for the feature pool. For the STRATH testbed, statistics such as maximum, minimum, average, variance, kurtosis, etc., are extracted for each processing phase. A trial of a complete black-box data-driven approach has been conducted for the STRATH testbed. For each phase, a three-level wavelet transform has been applied to the data. Statistical features have been extracted from each frequency band, resulting in more than 2,000 features. A simple feature selection algorithm, such as selecting the top *n* features with largest correlation with

the target CMM dimensions, provides an acceptable level of prediction although such a method is far from optimum as it ignores the relationships between features.

However, for some manufacturing applications, inside information of the machinery is valuable and should not be ignored. It can be used to extract features from time series which simple statistics may not cover. For example, when considering the detailed movement of the chuck head, the forging phase is further divided into 5 subphases as shown in Figure 7. The maximum of the first subphase (indicated by a red dot) is highly likely to be related to one of the CMM dimensions according to the mechanism of hammer movement.

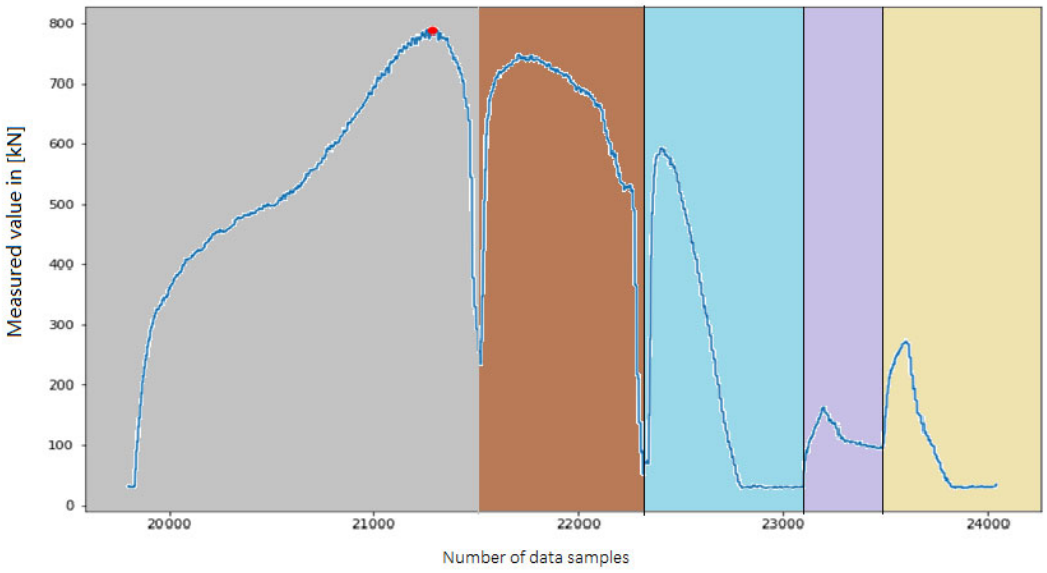

Figure 7. Subphases in the forging phase of a typical part.

The stage of feature selection can be combined or separated from that of regression or classification. The choice depends on the applied algorithms. Let the STRATH testbed be an example. It is possible to select features based on simple criteria of thresholding the correlation values between features and target CMM dimensions. It is also reasonable to put high a priori confidence on the features based on our understanding regardless of correlation value, e.g., temperature drop during the pick-up phase. There are algorithms which have feature selection integrated with regression and classification. Whether thresholding is needed is a choice depending on the number of features extracted and the computational complexity of the algorithms and other factors such as the training dataset size and the algorithm ability.

For the STRATH testbed, thresholding and algorithmic feature selection have both been carried out due to the large number of features from sensors. In thresholding, for example, those features whose correlation with the target dimension is greater than a certain threshold are selected to lower the number of features. Random Forest regressor and Bayesian Additive Regression Tree (BART) have been subsequently applied for the problem of CMM dimension prediction. Both algorithms give similar levels of prediction accuracy, although for different CMM dimensions one algorithm may be slight better than the other. However, the computational time for the BART algorithm is longer than that for the Random Forest algorithm. In a later section, an uncertainty-aware feature selection algorithm is discussed in more detail. If using the *ReliefF* algorithm, the prediction accuracy, in comparison, is much lower. The difference arises from the size of the training data size being limited for the STRATH testbed. The weights of a filtering type of algorithm, such as *ReliefF*, have not yet converged to the true

value with the training data, especially when the number of features in the candidate pool is large.

Therefore, for algorithm/model selection in regression and classification, the general principle, as in many other applications, involves consideration of accuracy, robustness, and computational simplicity. Accuracy is obviously an important indicator for algorithm selection. Robustness towards signals, measurement noise and numerical problems is also important. Computational simplicity, although becoming less crucial with current technology, is still valuable and enables potential real-time processing. As for a ML algorithm, explainable and trustworthy AI are desired. For the STRATH testbed, it is helpful to know what features the ML algorithm selects and if those features match our understanding. These are practical aspects for practitioners.

4 REDUNDANCY, MEASUREMENT COVERAGE AND ACCURACY

4.1 BACKGROUND

In modern factories there is often a quality target associated with a process output. This quality target is most expressive if it is quantitative, i.e., expressed as a numerical value, as opposed to being only qualitative. If the quality target can be defined as a measurable physical quantity, it can be formulated as a measurand in the metrological sense [VIM, 2008]. The symbol Y will be used to denote this measurand that is associated with the process output quality target. The sensors in the industrial sensor network can be seen as a collection of measuring instruments providing information about some input quantities. The set of input quantities measured by the sensor network will be denoted by X. If there exists a mathematical relationship f (e.g., a formula, an algorithm, or model equations) that relates Y to X, one can write $Y = f(X)$. If there exists at least one relationship f then the sensor network possesses (at least) minimum measurement coverage for the measurand Y. In section 4.2 the concept of a process output quality target and its relationship to different types of uncertainty is presented in more detail.

In some cases, it is desirable to have more than a minimum measurement coverage in the sensor network, e.g., to make the network more resilient against sensor node failures, or to decrease the uncertainty of the measurand. In this case, more sensors than are strictly needed can be employed, leading to a higher measurement coverage than is strictly needed. The sensor network is now called metrologically redundant, and there will be more than one way of deriving the value of the measurand from the measured input values. In section 4.3 some metrics to quantify the measurement coverage or redundancy will be presented. These metrics may be helpful, for example, for comparing different network designs.

The aspect of measurement accuracy and its relationship with redundancy will then be addressed. If the network is metrologically redundant, a smart combination of the input quantities can be used to improve the measurement accuracy, i.e., lower the measurement uncertainty, of the measurand. In section 4.4 it will be shown how this improvement can be achieved. In addition, some other usages of redundancy are presented in this section and section 4.5. Finally, in section 4.6 some of the ideas will be illustrated by applying them to industrial sensor network data provided by the STRATH testbed.

4.2 PROCESS OUTPUT QUALITY TARGET

In this section the concept *process output quality target* is studied in more detail. The process output can be described by a set of quantities associated with the parts created on the production line of the smart factory. One example could be a specific geometrical dimension (length, diameter, etc.) of a produced part. A *quality target* can be thought of as a (set of) requirement(s) on the (set of) quantity(ies) describing the process output. Denote $u(Y)$ as the

uncertainty associated with an estimate y of the quantity Y. Let the quantities y_1 and y_2 be the lower and upper bound of the quality target. The process output quality target for a single quantity can be formulated in many ways, e.g.:

a) Focus on value: Main requirement: $y_1 + u_0 \leq Y \leq y_2 - u_0$ with additional requirement $u(Y) \leq u_0$
b) Focus on uncertainty: Main requirement: $u(Y) \leq u_0$ with additional requirement $y_1 + u_0 \leq Y \leq y_2 - u_0$

Although formulations a) and b) are mathematically equivalent, the focuses in the quality target are different. In a), the underlying idea is that the value of main interest Y (e.g., the length of a produced part) should be inside tight tolerances and be measured with sufficiently low uncertainty. In b) the focus is rather on the uncertainty of Y, i.e. the value of Y may lie in a certain range but the most important requirement is that the value is known with a low uncertainty (e.g., the length of produced parts may vary but the exact length should be known). The measurement accuracy of the sensors in the network required to be able to meet the process output quality target depends on the model. More explicitly, for the measurand value y, the sensor values x and, some unobserved quantities (latent variables) w, one can write:

$$y = f(x, w)$$
$$u(y) = g(x, u(x), w, u(w))$$

Here, W are some non-measured random quantities (uncertainty sources), which are supposed to have zero mean, uncertainty $u(w)$ and non-measured values w. In the case of a linear model, the function g involves the partial derivatives of the model function f. Note that in the case of a strongly non-linear model with respect to the uncertainties, the value y could even depend on the uncertainties, i.e. $y = f(x, u(x), w, u(w))$. To understand the value of the measurand and its uncertainty, it is essential to know the functions f and g. Now, some qualitative statements regarding Y, $u(Y)$ and the models will be made.

The value of Y can be out of the specified range $[y_1 + u_0, y_2 - u_0]$ due to:

a) Value(s) of X being outside a desired range (e.g., the temperature of a forged part is too low)
b) Value(s) of W being outside a desired range (e.g., the humidity on which Y depends has unknown difference from a nominal value of humidity: note that $w = 0$ in the model and only additional measurements of W or Y can give this information)

The value of $u(Y)$ can be higher than required, $u(Y) > u_0$, due to:

c) Value(s) of $u(X)$ being higher than desired (e.g., the uncertainty of the temperature sensors is too large)
d) Value(s) of $u(W)$ being higher than desired (e.g., the uncertainty of the unknown humidity is too large)

Point c) addresses the measurement accuracy of the sensors.
Point d) addresses measurement coverage as a special case in the following way. A sub-optimal measurement coverage will result in a sub-optimal measurement model and higher $u(Y)$ than desired (e.g., temperature is only measured at one location and not completely representative for the whole part, leading to an increased measurement uncertainty, represented by a parameter W_1). In this case, increasing the measurement coverage and increasing the redundancy can lead to a lower measurement uncertainty $u(Y)$. To quantify this, additional models are needed to calculate the uncertainty in the case of a changed measurement coverage.

Note that, in general, meeting a quality target does not exclusively depend on measurement coverage and measurement accuracy. Also, whether variables are controlled or not (e.g.,

ambient temperature, play in machine axes, positioning accuracy of actuators, etc.) matters, and the values the variables can attain (e.g., the range of actuators) are an important aspect in being able to meet quality targets. Put differently, if every physical aspect (infinite measurement coverage) of a machine is measured and known with infinite accuracy, the machine may still not be able to produce specific parts according to the requirements (process output quality targets).

4.3 METHODS TO QUANTIFY MEASUREMENT COVERAGE AND REDUNDANCY

One aspect of *measurement coverage* is the amount of *metrological redundancy* in the industrial sensor network. Assessment of redundancy can be carried out using physical reasoning or can be data-driven, and different outcomes are possible. For example, two temperature sensors installed close to each other can be judged redundant from a physics-based point of view, but their measurement data can happen to be quite distinct. On the other hand, two temperature sensors installed far away from each other can be judged non-redundant from a physics-based point of view, but their measurement data can happen to be so similar that they are numerically judged as redundant measurements.

Various metrics can be defined in this context [Kok, 2020a] and they are all aimed at different aspects of measurement coverage. They address the number of replicated sensors, the relevance of single sensors for the measurand of interest, and the redundancy of the complete sensor network. In this guide no single 'best' or 'always good' practice or metric is proposed. What works for a given problem may not work for another problem. Various options will be given which the reader may try out for a specific problem and judge on their merits.

The metrics help to quantify different aspects of the measurement coverage. How much measurement coverage is needed to meet a process output quality target depends on the specific problem and associated mathematical model. This question cannot be answered in a general way.

4.3.1 Sensor replication

Sensor replication is the aspect that a(n approximate) functional relationship may exist between measured quantities. When a set of synchronized sensor data is organized as a matrix $\boldsymbol{X}$ with columns corresponding to individual sensors and rows to time stamps, the following metrics could be useful for quantifying the amount of sensor replication:

- Repl-Rank($\boldsymbol{X}$): calculate the matrix rank of $\boldsymbol{X}$
- Repl-Cond($\boldsymbol{X}$): calculate the condition number of $\boldsymbol{X}$
- Repl-Clust($\boldsymbol{X}$): apply a clustering algorithm to the columns of $\boldsymbol{X}$, e.g., identify clusters based on the Pearson correlation coefficients between pairs of columns.

Uncertainty aware versions of these metrics can be created by weighting $\boldsymbol{X}$ by the inverse of the lower Cholesky factor of the uncertainty matrix $U_{\boldsymbol{X}}$ (i.e., if $U_{\boldsymbol{X}} = L^{\mathrm{T}} L$, with L a lower triangular matrix, apply the metrics to $\boldsymbol{X'} = L^{-1}\,\boldsymbol{X}$).

4.3.2 Sensor relevance

Sensor relevance is the concept of how much the quantity X_i measured by sensor i contributes to knowing the value and uncertainty of the measurand Y. Possible metrics for quantifying the sensor relevance are:

- Rel-SensCoef(X_i, Y): calculate the sensitivity coefficient $c_i = \mathrm{d}f/\mathrm{d}x_i$ and evaluate $|c_i|u(x_i)/u(y)$.

- Rel-PearCor(X_i, Y): calculate the Pearson correlation coefficient between known values of Y and corresponding measurement data X_i of sensor i.

4.3.3 Metrological network redundancy

Metrological network redundancy is the aspect that the measurand can be derived in multiple, independent (or rather not fully correlated) ways from the sensor data. It can be quantified by the following metrics:

- Red-Excess(X, Y, u): Number of sensors present in the network in excess of the minimum number necessary to determine values of the measurand with at most uncertainty u. Here, if Red-Excess(X, Y, u) = m, then any m arbitrary sensors can be removed from X, and still it is possible to determine Y, with $u(Y) \leq u$.
- Red-Unc(X, Y, m): Maximum uncertainty increase in Y when m arbitrary sensors are removed from the set of sensors given by X, compared to the case of using all sensors.

4.4 METHODS TO IMPROVE MEASUREMENT ACCURACY BASED ON REDUNDANCY

As discussed in sections 4.1 and section 4.2, redundancy can be used to lower the measurement uncertainty $u(Y)$ and thereby contribute to reaching the process output quality target. Redundancy is essentially a desirable property of the network. However, if there are multiple estimates of the same quantity, one may ask oneself, which value should be used. Also, for various algorithms, e.g., a linear regression fit, linear dependence of the input quantities is undesirable and should be removed beforehand. In the following section, some ways of using redundancy to lower the uncertainty of the estimate of interest and to remove linear dependencies from the data are presented.

4.4.1 Uncertainty weighted averaging

In a metrological setting the common approach is to use the uncertainty weighted average. For a vector of n sensor values $\boldsymbol{x} = [x_1, x_2, \ldots x_n]^T$ all independently estimating the same quantity X_0, with individual sensor uncertainties $u(x_i)$, $1 \leq i \leq n$. The best estimate x_0 of X_0 with uncertainty $u(x_0)$ is given by

$$u(x_0) = \left(\sum_{i=1}^{n} \frac{1}{u^2(x_i)} \right)^{-1/2},$$

$$x_0 = u^2(x_0) \sum_{i=1}^{n} \frac{x_i}{u^2(x_i)}.$$

In the case that the sensors are not independent, and their joint covariance matrix is given by $U_{\boldsymbol{x}}$, then the best estimate and uncertainty are given by

$$u(x_0) = (\boldsymbol{e}\, U_{\boldsymbol{x}}^{-1} \boldsymbol{e})^{-1/2},$$
$$x_0 = u^2(x_0)\, \boldsymbol{e}\, U_{\boldsymbol{x}}^{-1} \boldsymbol{x},$$

where $\boldsymbol{e} = [1, \ldots, 1]^T$ is of suitable dimension (in the expression above n x 1, below it is k x 1). In an even more general case, there are k linear equations to derive the scalar X_0 from the vector $\boldsymbol{X}$, i.e., there is a k-dimensional linear system of the form

$$X_0\, \boldsymbol{e} = \boldsymbol{a} + A\, \boldsymbol{X},$$

where $\boldsymbol{a}$ is a column vector of dimension k x 1 and A is a matrix of dimension k x n. This system indicates the relationship between the quantities in the absence of nuisance variables. If A and $U_{\boldsymbol{x}}$ have full rank, and the measured sensor values are given by $\boldsymbol{x}$ then the best estimate x_0 of X_0 with associated uncertainty $u(x_0)$ follows from

$$\begin{aligned} \boldsymbol{x}_0 &= \boldsymbol{a} + A\,\boldsymbol{x}, \\ U_{\boldsymbol{x}_0} &= A\,U_x\,A^{\mathrm{T}}, \\ \boldsymbol{v} &= \left(\boldsymbol{e}^{\mathrm{T}}\,U_{\boldsymbol{x}_0}^{-1}\,\boldsymbol{e}\right)^{-1}\,U_{\boldsymbol{x}_0}^{-1}\,\boldsymbol{e}, \\ x_0 &= \boldsymbol{v}^{\mathrm{T}}\,\boldsymbol{x}_0, \\ u(x_0) &= \sqrt{\boldsymbol{v}^{\mathrm{T}}\,U_{\boldsymbol{x}_0}\,\boldsymbol{v}}. \end{aligned}$$

If A and/or $U_{\boldsymbol{x}}$ are rank deficient, a more complex algorithm needs to be applied to remove this degeneracy. This algorithm has been described in [Kok, 2020b] and [Eichstädt, 2021].

This approach treats the incoming, synchronized data time point per time point, or row by row in terms of a data matrix $\boldsymbol{X}$. The data are not explicitly treated as time series (and do not need to be time series). It is also assumed that there is uncertainty information available.

4.4.2 Principal Component Analysis

Another option to remove redundancy from the data is to use Principal Component Analysis (PCA) [Wold, 1987], which can also be made uncertainty aware [Görtler, 2019]. This approach fully concentrates on the data matrix X and disregards the measurand Y. If the sensor values are expressed in different units, it is common to centre and normalize the data to have zero mean and unit variance. This is not needed if all values have the same unit. The result of the PCA analysis can then be used to reduce the dimensionality of the data. However, as the procedure disregards the measurement model and the measurand Y, it could happen that relevant information regarding the measurand is removed. For a data driven approach aimed at creating a data model, this technique can be used. In the case a measurement model is present, this method might not be most suitable.

4.4.3 Identifying suspicious values

There are many different methods in the literature for identifying suspicious values or outliers. In a metrological context the sensor values should be judged in an analysis incorporating the instrument uncertainty and expected variation of the measured process, rather than solely based on a purely data driven, statistical analysis. In the case of redundancy, multiple independent estimates of the measurand or of an intermediate quantity are present. If the uncertainties of the sensor follow a multivariate normal distribution, a chi-squared test can be used to determine if all estimates are consistent with respect to the model. For this test, the quantity χ^2_{obs} is computed according to

$$\chi^2_{\mathrm{obs}} = (\boldsymbol{x}_0 - x_0\boldsymbol{e})^{\mathrm{T}}\,U_{\boldsymbol{x}_0}^{-1}\,(\boldsymbol{x}_0 - x_0\boldsymbol{e}).$$

The quantity χ^2_{obs} possesses a chi-squared distribution χ^2_{n-1} with $n - 1$ degrees of freedom. Now select a probability p for the consistency evaluation, e.g., p = 5 %, which corresponds to the probability of falsely rejecting the data as not being generated by the assumed model, when it actually is generated by the model. The observed value χ^2_{obs} should now be compared with the 1 – p quantile $q_{n-1,1-p}$ of χ^2_{n-1} and when $\chi^2_{\mathrm{obs}} > q_{n-1,1-p}$, the data should be marked as inconsistent or suspicious. A possible solution is now to remove the sensor values one by one, and to calculate if a consistent solution can be found. If there are multiple possibilities,

keep the one with the lowest value of χ^2_{obs}. If there are none, continue with removing any combination of $k > 1$ sensors until a solution is found. In the case there is no intermediate system of equations, i.e., $\boldsymbol{x}_0 = \boldsymbol{x}$, this procedure is called the Largest Consistent Subset (LCS) algorithm. This approach is more widely used in metrology, e.g., for the evaluation of intercomparisons [Cox, 2002]. In the case of an intermediate system of equations $\boldsymbol{x}_0 = \boldsymbol{a} + A\,\boldsymbol{x}$, this procedure is called the Largest Consistent Subset of Sensors (LCSS) algorithm [Kok, 2002b; Eichstädt, 2021].

4.5 DATA QUALITY VERIFICATION

Another important use of redundancy is to verify the quality and correctness of the data. Some general approaches that can be used depending on the quantities measured by the network are:

- Verification of the correspondence between nominal and actual values
- Verification of a linear dependence in the data
- Verification of a general functional dependence in the data
- Verification of a differential and / or integral dependence in the data
- Verification of a synchronization time point

The idea each time is that based on a physical understanding of the system, or based on experience of operating the sensor network, certain mathematical relationships should exist between certain sensor values or time series of certain sensors. The algorithm will check for each new batch of sensor data if these constraints are satisfied. If this is not the case, a warning will be raised, and the user is advised to verify the quality of the data in more detail.

4.6 NUMERICAL EXAMPLE: STRATH TESTBED

Theoretical aspects of redundancy evaluation from a metrological angle are discussed above. There is often a little gap when applying the model to a practical system, especially when a complex manufacturing process is involved. In this section some of the introduced metrics will be applied to a dataset obtained from the STRATH testbed as explained in section 1.3, [Tachtatzis, 2019]. The measurands in the STRATH testbed are the CMM dimensions after forging. The input variables $\boldsymbol{X}$ ideally are sensor data. But sensor data are time series and the functional mapping f is difficult if not impossible to establish. However, the idea of redundancy evaluation is still applicable.

To verify the level of useful information contained in data, features are extracted from the signals. In Figure 8, the Repl-Cond metric is applied, quantifying the sensor replication by means of the condition number. Five parts have a lower value than the others. This might indicate a change in the process, or some change related to the data processing.

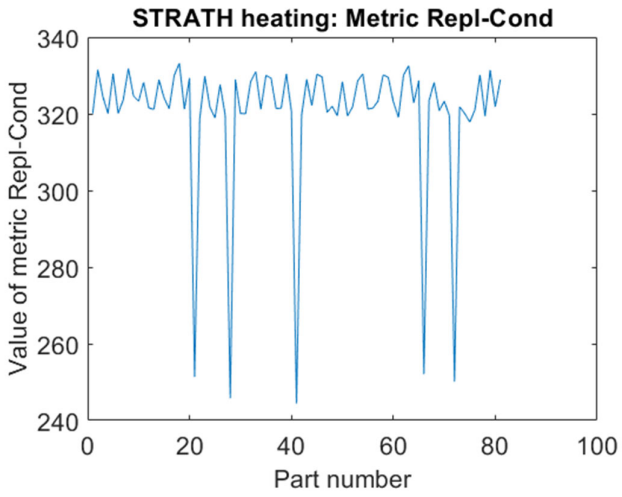

Figure 8: Metric Repl-Cond applied to heating sensors of the STRATH testbed dataset.

In Figure 9, the metric Rel-PearCor is calculated for the forging sensors, quantifying the relevance of each sensor for the part dimensions using the Pearson correlation coefficient and a particular feature of the data. None of the sensors is very informative for any of the dimensions if this feature is used, as the metric value lies always below 0.6.

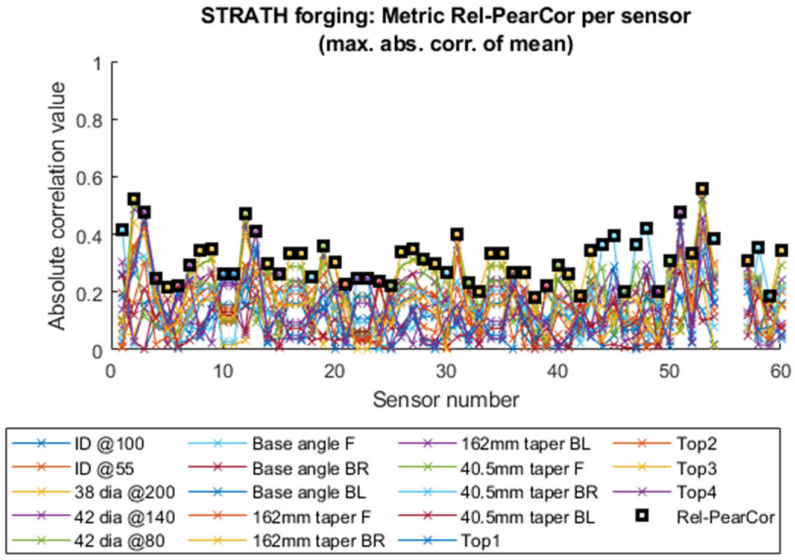

Figure 9: Metric Rel-PearCor applied to forging sensors of the STRATH testbed dataset.

In Figure 10 the nominal and measured values of the speed at the so-called L-axis is shown for a single part. As nominal and measured values lie closely together, there is no indication of any malfunction which could eventually endanger meeting the process output quality target. On the right, the values of the actual speed and the axis output percentage at the same axis are shown for the same part. One output is clearly a scaled version of the other output, which constitutes a form of redundancy. A data verification algorithm could check for this relationship and raise a warning if this condition is not met anymore. If uncertainties are provided for these sensor readings, the approach presented in section 4.4 can be used to take advantage from

the redundancy by calculating an aggregated estimate with lower uncertainty, and subsequently removing this redundancy from the data for further processing.

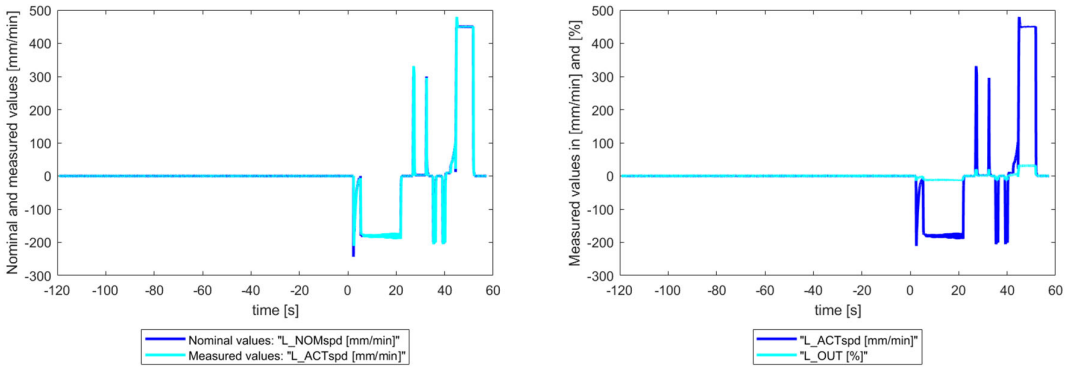

Figure 10: (Left) Nominal and measured values of speed of the hammers at the L-axis for part 50. (Right) Values of actual speed and axis output percentage at the L-axis for part 50. One output is clearly a scaled version of the other output.

Figure 11 shows an example of redundancy where there is a differential or integral dependence in the data. Both speed and position are returned by the sensor system. A data verification check can be made by integrating the speed values and comparing with the position and raising a warning if the residuals lie outside a guard band ('allowed tolerance').

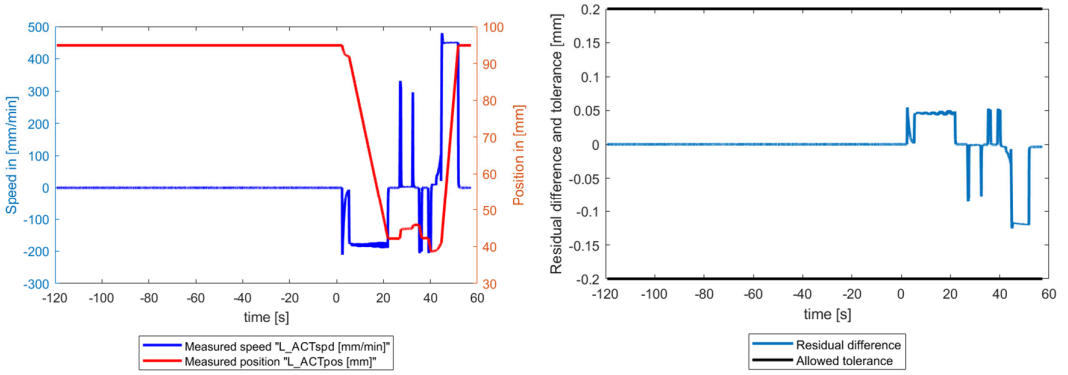

Figure 11: (Left) Measured speed and position of the hammers at the L-axis for part 50. (Right) Residuals and guard band ('allowed tolerance') for integrating the measured speed and comparing it with measured position for part 50.

5 FEATURE EXTRACTION AND SELECTION

5.1 BACKGROUND

As the volume of data increases exponentially so the quality of data required for processing by ML algorithms decreases only gradually. High-dimensional datasets lead to the prevalence of noisy and irrelevant data that can cause overfitting of the model and affect the performance of the learning algorithm. Moreover, high-dimensional datasets require high computing power and large volumes of storage space and, hence, there is a need to select a representative subset of features to address these challenges. Feature selection is also particularly important

in the context of small sample classification/regression problems when the number of available training samples is very small compared to the number of features.

In their review paper on feature selection methods for high-dimensional data, [Gnana, 2016] classify the process of feature selection into feature subset selection and feature ranking methods based on how the features are combined for evaluation. Feature selection methods are further divided into filter, wrapper, embedded and hybrid methods based on how the supervised learning algorithm is employed in the feature selection process.

Filter methods start with all features and select the best feature subset based on statistical measures such as Pearson's correlation, Fischer score, Linear Discriminant Analysis, ANOVA, Chi-square, Wilcoxon Mann Whitney test and Mutual Information. The best example for the feature subset-based method is correlation-based feature subset selection [Hall, 1999]. Filter methods are fast, efficient, and independent of the learning algorithm, yet they do not account for the correlation between features and between classifiers.

Wrapper methods select features based on inductive algorithms, meaning that the learning algorithm is included in the validation of the generated feature subsets (a black box approach). Commonly used wrapper methods are Recursive Feature Elimination, Sequential Feature Selection algorithms [Aha, 1996] and Genetic Algorithms. Wrapper methods account for the dependencies between features and are more accurate than filter methods. However, due to the repeated learning steps and cross-validation, such methods are computationally more complex. Moreover, since wrapper methods iterate to evaluate the selected feature subset, some features may be wrongly dropped at the initial stage or searching overhead may occur. Embedded methods perform appropriate feature selection and model learning at the same time, and the features are selected during the training stage of the model. Among the available techniques, sparse Bayesian Machine Learning algorithms are mainly used: Relevant Sample-Feature Machine (RSFM), Variational Relevant Sample-Feature Machine (VRSFM), Relevant Vector Machine (RVM) [Tipping, 2001] or Relevant Group Selector (RGS) [Subrahmanya, 2010]. Embedded methods are computationally more efficient than wrapper methods and more accurate than the two other methods, yet they offer poor generalizability due to their dependence on the learning algorithm.

Hybrid methods were originally proposed to combine the best properties of filters and wrappers. First, a filter method is used in order to reduce the feature space dimension, possibly obtaining several candidate subsets. Then, a wrapper finds the best candidate subset. Hybrid methods usually achieve high accuracy that is characteristic to wrappers and high efficiency that is characteristic to filters [Jović, 2015].

Embedded methods offer a good trade-off between complexity and accuracy [Venkatesh, 2019] and for that reason, they are considered more suited to the feature selection task in industrial applications especially if one intends to account for feature uncertainty. The following sections focus on RVM and RGS algorithms.

5.2 RVM ALGORITHM

Let $\mathcal{D}$ be the training set, such that $\mathcal{D} = \{(X_i, t_i), i = 1, \dots, n\}$, n_s the number of sensors, s_k the feature set of sensor k, d_k the number of features extracted from sensor k, d the total number of extracted features $\left(d = \sum_{k=1}^{n_s} d_k\right)$, X_i the input vector storing the ith observation from the whole set of sensors, and $\phi(X_i) \in R^d$ the d-dimensional feature vector extracted from the input X_i.

The output response t_i is representative of the true model y_i with the addition of noise ϵ_i such that the output response can be written as a simple additive model:

$$t_i = y_i + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \tau^{-1}),$$

where y_i can now be modelled as a function of the computed features and it is assumed here that the relationship between the true model output can be represented as a linear combination of the feature vector and a weight vector w:

$$y_i = w^{\mathrm{T}}\phi(X_i).$$

RVM uses a zero-mean Gaussian prior over w, hence, there is an individual hyper-parameter $\lambda_{1,j}$ associated with each w_j ($\lambda_{1,j}$ stands for the inverse variance of the j-th parameter). With such a formulation, relevant features are selected according to the value of the estimate of $\lambda_{1,j}$: high values of the estimates of $\lambda_{1,j}$ will relate to irrelevant features (the variance of the weight w_j will tend to 0) whereas, on the contrary, low values of the estimates of $\lambda_{1,j}$ will lead to relevant features. The model assumes a conjugate hyper prior for both $\lambda_{1,j}$ (here the inverse variance of the weight w_j is controlled by a Gamma prior) and τ (the inverse variance of the noise ϵ is controlled by a Gamma prior). The complete model specification is given below.

Likelihood function:

$$p(t \mid X, w, \tau) = \prod_{i=1}^{n} \mathcal{N}\left(t_i \mid w^{\mathrm{T}}\phi(X_i), \tau^{-1}\right)$$

Conjugate weight prior:

$$p(w \mid \lambda_1) = \prod_{j=1}^{d} \mathcal{N}\left(w_j \mid 0, \lambda_{1,j}^{-1}\right)$$

Conjugate hyper-parameter prior:

$$p\left(\lambda_{1,j} \mid \alpha, \beta\right) = \Gamma\left(\lambda_{1,j} \mid \alpha, \beta\right)$$

Conjugate hyper-prior for the inverse noise variance:

$$p(\tau) = \Gamma(\alpha_\tau, \beta_\tau)$$

The posterior probability, $p(w, \lambda_1, \tau \mid t)$, over all the unknown parameters, given the data, is expressed according to Bayes' rule as:

$$p(w, \lambda_1, \tau \mid t) = \frac{p(t \mid X, w, \lambda_1, \tau)p(w, \lambda_1, \tau)}{p(t \mid X)}$$

5.3 RGS ALGORITHM

[Subrahmanya, 2010] proposed a hierarchical Bayesian formulation that allows to introduce grouping (features from different sensors will correspond to a different group) and to simultaneously perform feature selection within groups to reduce over-fitting of the data. Other useful information that can be obtained from such an algorithm is a rank order among the selected sensors based on the model parameter estimates. RGS is an extension of RVM that includes sensory grouping information.

The RGS model creates grouping information based on the sensors, such that w takes the form $w = \left\{w^{[1]}, w^{[2]}, \cdots, w^{[k]}, \cdots, w^{[n_s]}\right\}$ where each group represented by $w^{[k]}$ is associated with an extra hyper parameter $\lambda_{2,k}$. Following the same principle as for the RVM model, RGS

leads to relevant sensors as well as relevant features within sensor groups according to the values of the estimates of $\lambda_{2,k}$ and $\lambda_{1,j}$, respectively. The model can be summarized as follows.

Likelihood function:

$$p(t \mid X, w, \tau) = \prod_{i=1}^{n} \mathcal{N}\left(t_i \mid w^{\mathrm{T}} \phi(X_i), \tau^{-1}\right)$$

Conjugate weight prior:

$$p(w \mid \lambda_1, \lambda_2) = \prod_{k=1}^{n_s} \prod_{j \in s_k} \mathcal{N}\left(w_j \mid 0, \left(\lambda_{1,j} + \lambda_{2,k}\right)^{-1}\right)$$

Conjugate hyper-parameters prior:

$$p\left(\lambda_{1,j} \mid \alpha_1, \beta\right) = \Gamma\left(\lambda_{1,j} \mid \alpha_1, \beta\right)$$
$$p\left(\lambda_{2,k} \mid \alpha_2, \beta\right) = \Gamma\left(\lambda_{2,k} \mid \alpha_2, \beta\right)$$

Conjugate hyper-prior for the inverse noise variance:

$$p(\tau) = \Gamma(\alpha_\tau, \beta_\tau)$$

The posterior probability, $p(w, \lambda_1, \lambda_2, \tau \mid t)$, over all the unknown parameters, given the data, is expressed according to Bayes' rule as:

$$p(w, \lambda_1, \lambda_2, \tau \mid t) = \frac{p(t \mid X, w, \lambda_1, \lambda_2, \tau) p(w, \lambda_1, \lambda_2, \tau)}{p(t \mid X)}$$

5.4 ACCOUNTING FOR FEATURE UNCERTAINTY

We now want to account for feature sample uncertainty and this can be done by replacing the current error-free feature vector $\phi(X_i)$ for the i-th observation by a multivariate normal vector, $\Phi(X_i)$, given as:

$$\Phi_i \sim \mathcal{N}_d\left(\phi(X_i), \Sigma_\phi\right)$$

where $\Sigma_\phi = \mathrm{diag}\left(\sigma^2_{\phi_1}, \ldots, \sigma^2_{\phi_j}, \ldots, \sigma^2_{\phi_d}\right)$ and $\sigma^2_{\phi_j}$ represents the j-th feature known variance. Note that he covariance terms $\left(\sigma^2_{\phi_{ij}}\right)_{i \neq j}$ are all set to zero in this formulation.

The updated likelihood function and conjugate weight prior for the RVM model is then given by:

$$p(t \mid X, w, \tau) = \prod_{i=1}^{n} \mathcal{N}(t_i \mid w^T \Phi_i, \tau^{-1})$$
$$p(w \mid \lambda_1, \lambda_2) = \prod_{k=1}^{n_s} \prod_{j \in s_k} \mathcal{N}\left(w_j \mid 0, \sigma^2_{\phi_j} + \left(\lambda_{1,j} + \lambda_{2,k}\right)^{-1}\right)$$

5.5 PRIOR MODELLING

In most practical applications, it can be very difficult to properly elicit a prior distribution. In this case, the chosen prior distribution might be one which keeps the mathematics simple whilst

also assuming a large variance. Alternatively, a prior can assume equally likely values to represent complete prior ignorance. When using sparse Bayesian modelling (such as the RVM or RGS models presented above), priors are made non-informative by setting small values for the Gamma distribution parameters for λ_1 and λ_2 as well as for the inverse noise variance τ. A simple choice is to set $\alpha_1 = \alpha_2 = \beta = \alpha_\tau = \beta_\tau = c$, where c is small ($c = 10^{-6}$ in [Subrahmanya, 2010]). Prior modelling is a very important aspect of the Bayesian framework and many other choices could have been made for these priors, especially when expert knowledge is available. As an example, expert opinion might be that some values in a given range are more likely to occur than others.

5.6 INFERENCE

In practice, the posterior distribution cannot be obtained as a closed form distribution because the marginal likelihood $p(t \mid X)$ is intractable due to the well-known limitations of numerical integration techniques for high-dimensional integrals. For that reason, solving such problems resorts to approximate techniques.

5.6.1 Methods

To address intractable models in the field of Bayesian inference, several approximate techniques exist. [Subrahmanya, 2010] rely on the Empirical Bayes approach [Casella, 1985] to infer the model parameters where the prior distribution is estimated from the data and not integrated out as for the fully Bayesian treatment of a hierarchical model.

Bayesian programming is nowadays efficient to address such problem by using Markov Chain Monte Carlo (MCMC) algorithms (Metropolis–Hastings, Hamiltonian Monte Carlo, NUTS [Hoffman, 2014], Sequential Monte Carlo, etc.) and Variational Inference algorithms (Stein Variational Gradient Descent [Liu, 2016], etc.). Instead of drawing samples from the posterior, Variational Inference algorithms fit a distribution to the posterior making the initial sampling problem an optimization problem much faster to solve (the optimization problem generally consists in maximizing the Evidence Lower Bound or ELBO). Thus, mini batch based Variational Inference techniques drastically reduce the time needed to infer the model parameters compared to MCMC sampling techniques. Indeed, MCMC methods fail to converge in a reasonable amount of time as the number of dimensions rises.

5.6.2 MCMC chain convergence checking

Valid inferences from sequences of MCMC samples assume that the samples are derived from the true posterior distribution. However, the convergence rate of MCMC algorithms cannot be computed analytically in general, and so, in practical situations, it becomes necessary to determine the number of samples needed to ensure a good approximation to the target posterior density. Since the choice of such thresholds is problem specific, convergence diagnostic tools are used to verify convergence. Spectral analysis [Geweke, 1992] or Gelman-Rubin statistics [Gelman, 1992] are commonly used tools to perform the verification.

5.7 SIMULATED EXAMPLE

The regression example is taken from [Subrahmanya, 2010]. Six independent features $(f_1, f_2, f_3, f_4, f_5, f_6)$ were generated by sampling from six independent normal distributions with zero mean and unit standard deviation. A number of irrelevant features were generated by sampling from other independent normal distributions in order to simulate 10 sensor-feature groups. A total of 20 features per sensor was used in this case. The output y was generated such that it depends on all these six features using the following equation:

$$y = 0.2f_1 + 0.24f_2 + 0.28f_3 + 0.32f_4 + 0.36f_5 + 0.4f_6 + \epsilon$$

where ϵ denotes a vector of Gaussian noise with zero mean and standard deviation of 0.05. Figure 12 shows 320 samples of the output response y that will be used to train the model.

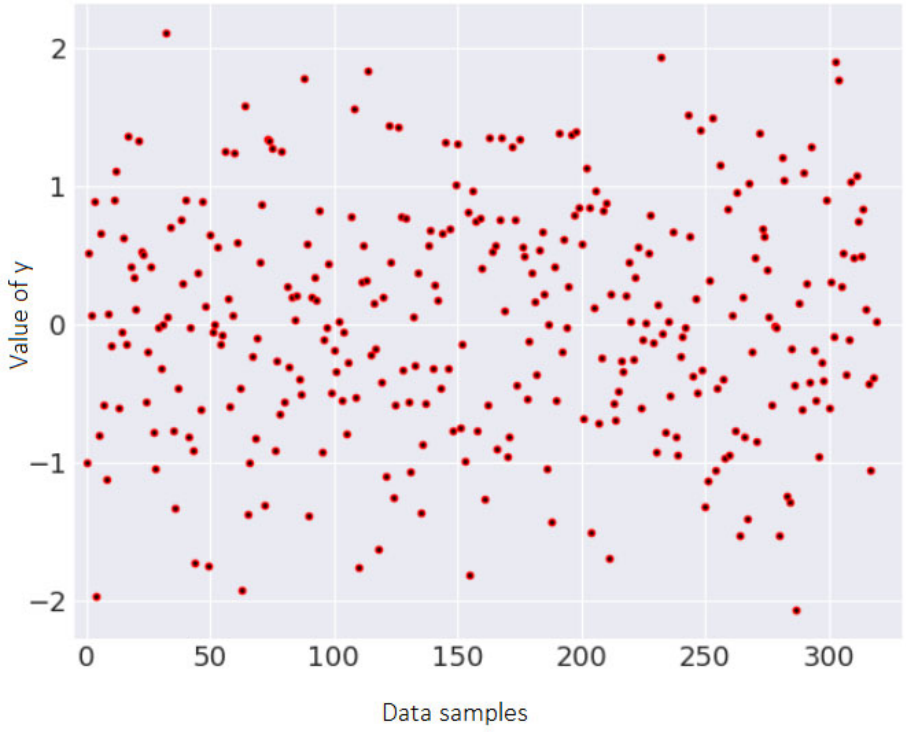

Figure 12: Samples from the output response y.

Figure 13 displays a boxplot of the 10 sensor-feature groups used to draw the training samples in the above figure. From this figure, we can clearly identify the relevant features $(f_1, f_2, f_3, f_4, f_5, f_6)$ as they are sampled from standard normal distributions while irrelevant features are sampled from normal distributions with higher dispersion (here, 10). The features to be selected in this example (a total of 17 features) are listed below:

- sensor 1 : f_1, f_2, f_3
- sensor 2 : f_3, f_4, f_5
- sensor 3 : f_1, f_2
- sensor 4 : f_3, f_4
- sensor 5 : f_2, f_3
- sensor 6 : f_4
- sensor 7 : f_5
- sensor 8 : f_4
- sensor 9 : f_5
- sensor 10 : f_6

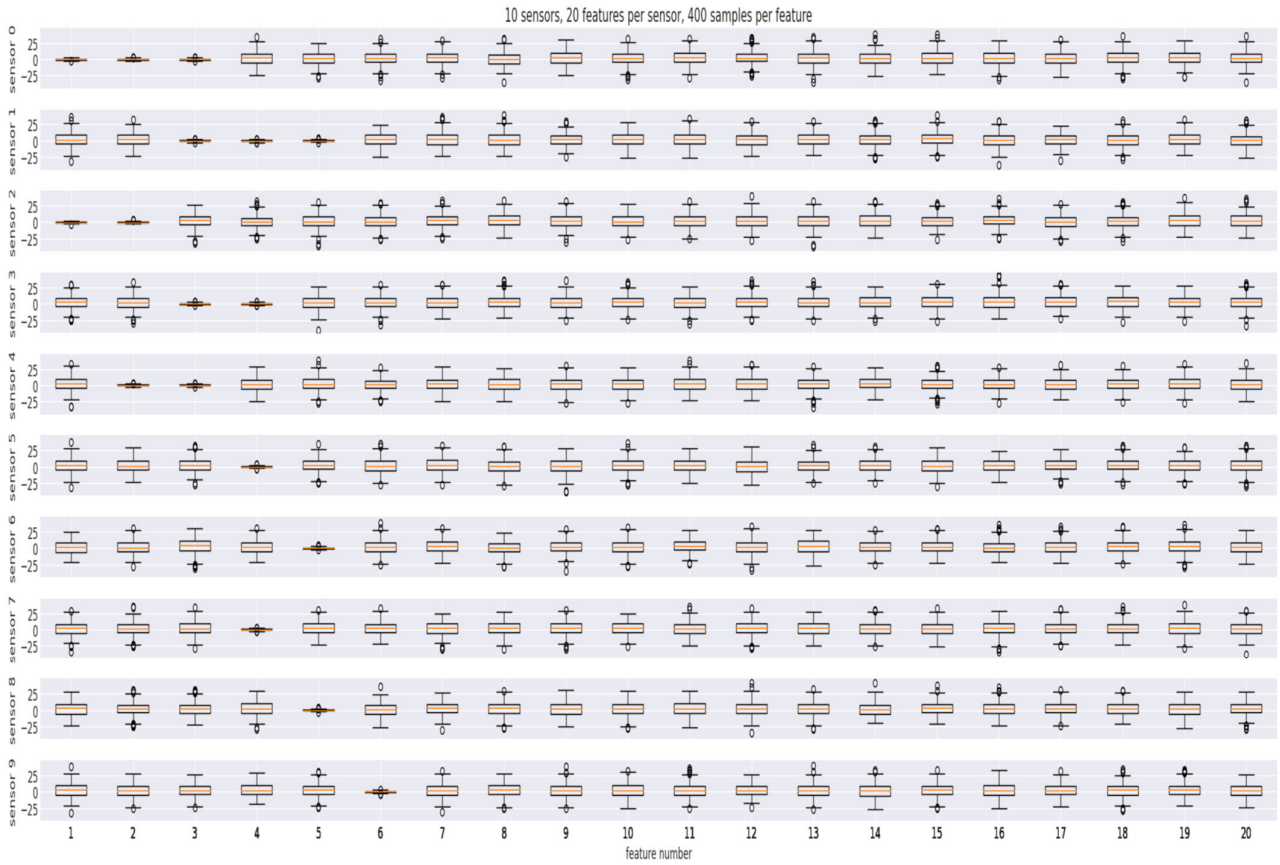

Figure 13: Feature sensor level representation.

5.7.1 Uncertainty-free features

Figure 14 shows the results of the feature selection for each sensor in the case of uncertainty-free features, and these correspond to the correct groupings listed above.

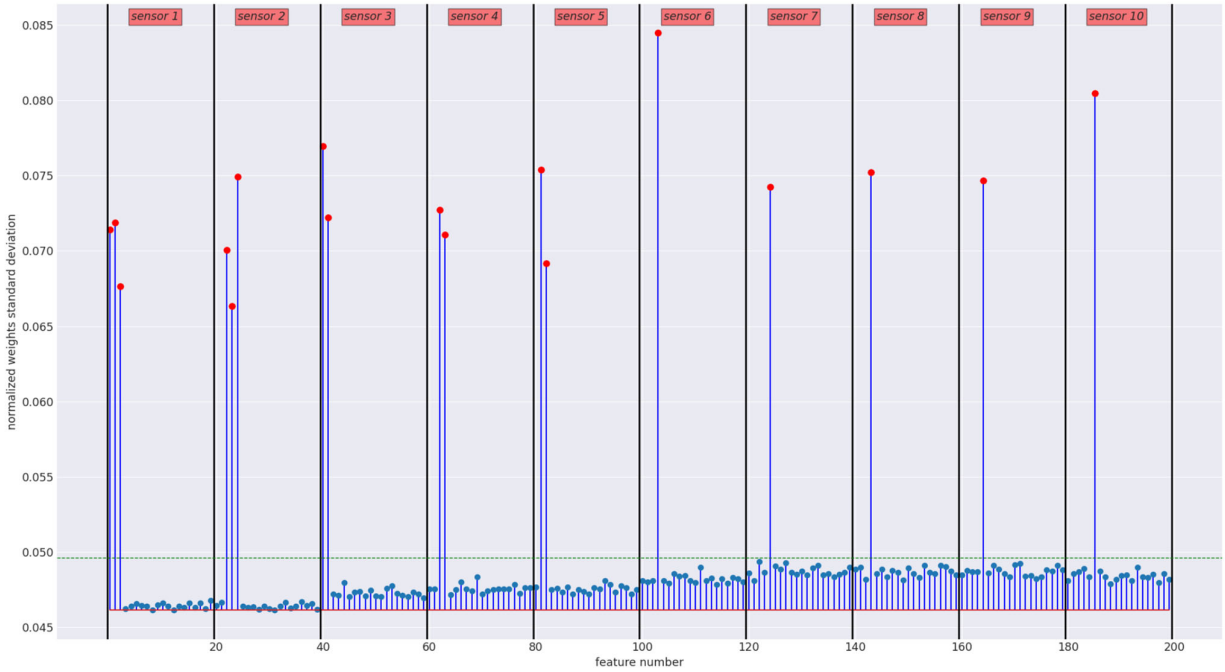

Figure 14: Selected features based on normalized weight standard deviations (uncertainty-free features). The elbow method is used to find the threshold (shown as the horizontal green dotted line).

5.7.2 Uncertain features

First (Figure 15), consider the synthetic regression example accounting for homogenous feature sample uncertainty by setting $\sigma_{\phi_j} = 0.2, \forall j$.

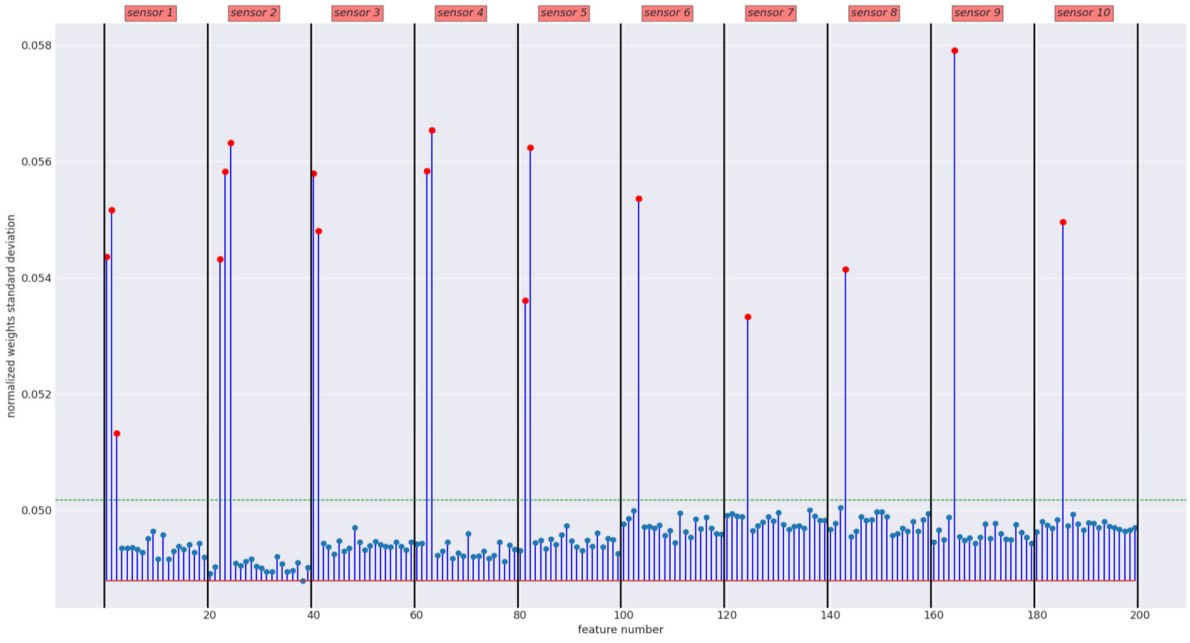

Figure 15: Selected features based on normalized weight standard deviations (homogeneous feature uncertainty with $\sigma_{\phi_j} = 0.2, \forall j$).

Second (Figure 16), consider the synthetic regression example accounting for heterogeneous feature sample uncertainty by sampling each σ_{ϕ_j} from a uniform distribution $\sigma_{\phi_j} \sim \mathrm{U}(0.1, 0.5)$.

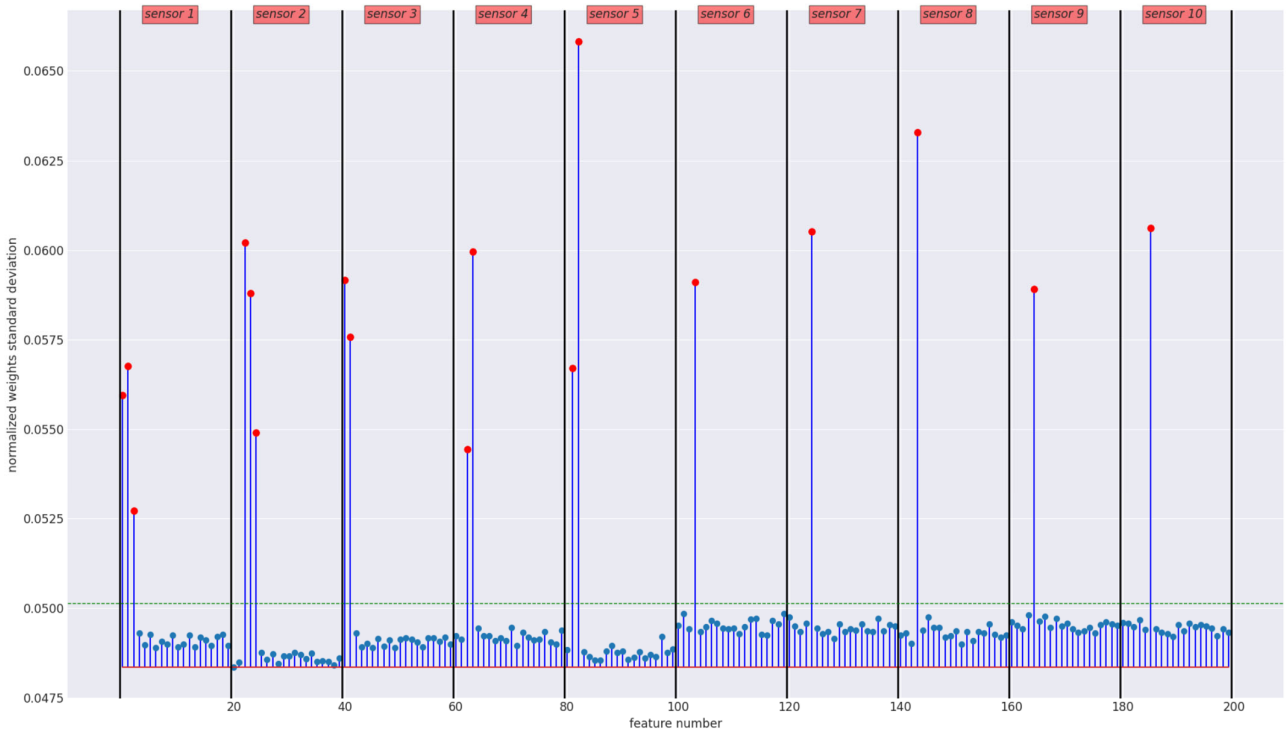

Figure 16: Selected features based on normalized weight standard deviations (heterogeneous feature uncertainty with $\boldsymbol{\sigma_{\phi_j} \sim U(0.1, 0.5)}$).

5.7.3 Uncertain features combined with noisy measurements

In this section, we investigate the robustness of the feature selection model against two noise levels: $\epsilon \sim \mathcal{N}(0, 0.1)$ and $\epsilon \sim \mathcal{N}(0, 0.5)$, for the two test cases (homogeneous and heterogeneous feature uncertainty) previously described. First (Figure 17), consider the synthetic regression example accounting for homogeneous feature sample uncertainty and noisy measurements by setting $\sigma_{\phi_j} = 0.2, \forall j$, at two noise levels: $\epsilon \sim \mathcal{N}(0, 0.1)$ and $\epsilon \sim \mathcal{N}(0, 0.5)$.

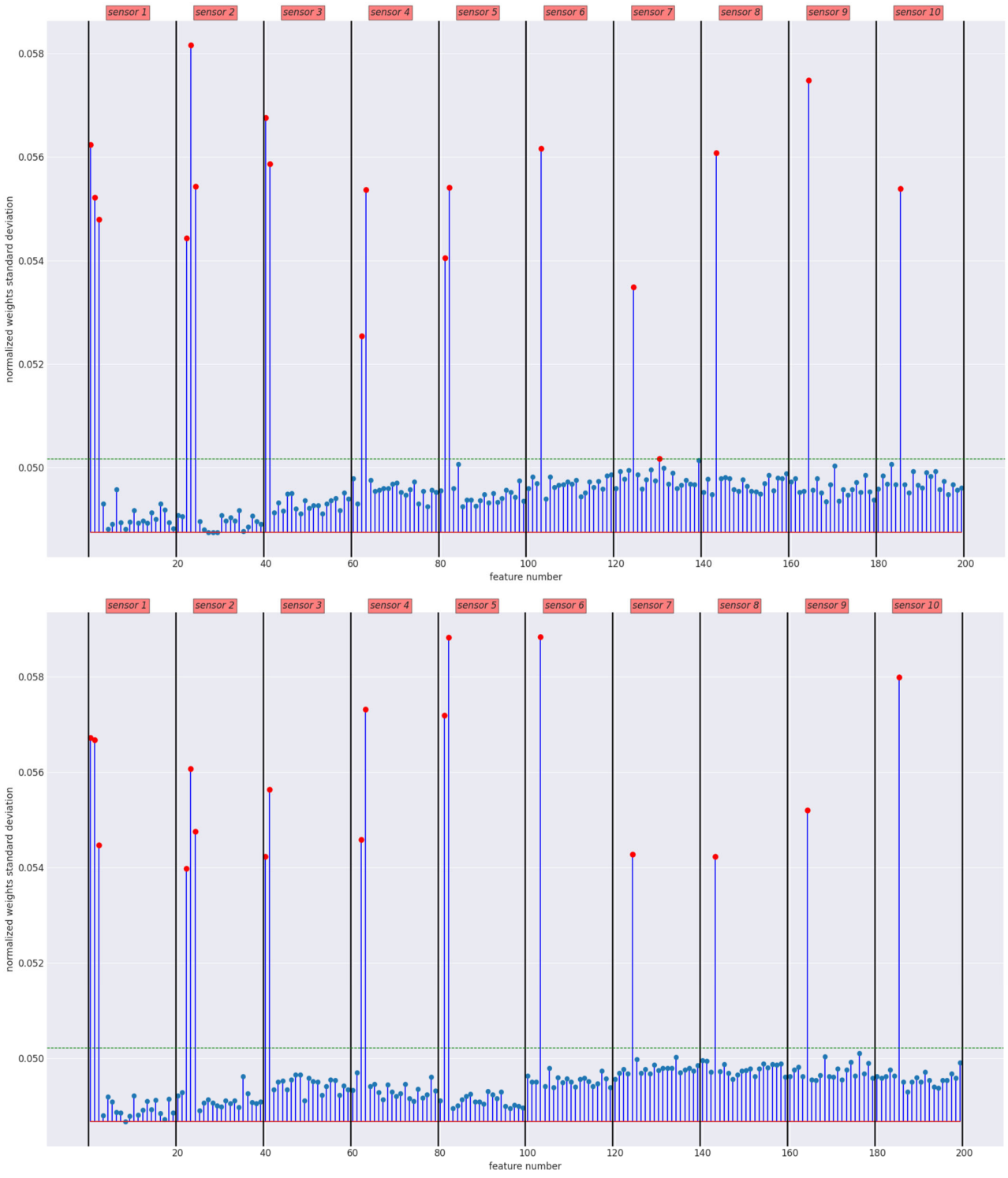

Figure 17: Selected features based on normalized weight standard deviations (homogeneous feature uncertainty with $\sigma_{\phi_j} = 0.2, \forall j$, at two noise levels: $\epsilon \sim \mathcal{N}(0, 0.1)$ (top figure) and $\epsilon \sim \mathcal{N}(0, 0.5)$ (bottom figure)).

Second (Figure 18) , consider the synthetic regression example accounting for heterogeneous feature sample uncertainty by sampling each σ_{ϕ_j} from a uniform distribution $\sigma_{\phi_j} \sim \mathrm{U}(0.1, 0.5)$ at two noise levels: $\epsilon \sim \mathcal{N}(0, 0.1)$ and $\epsilon \sim \mathcal{N}(0, 0.5)$.

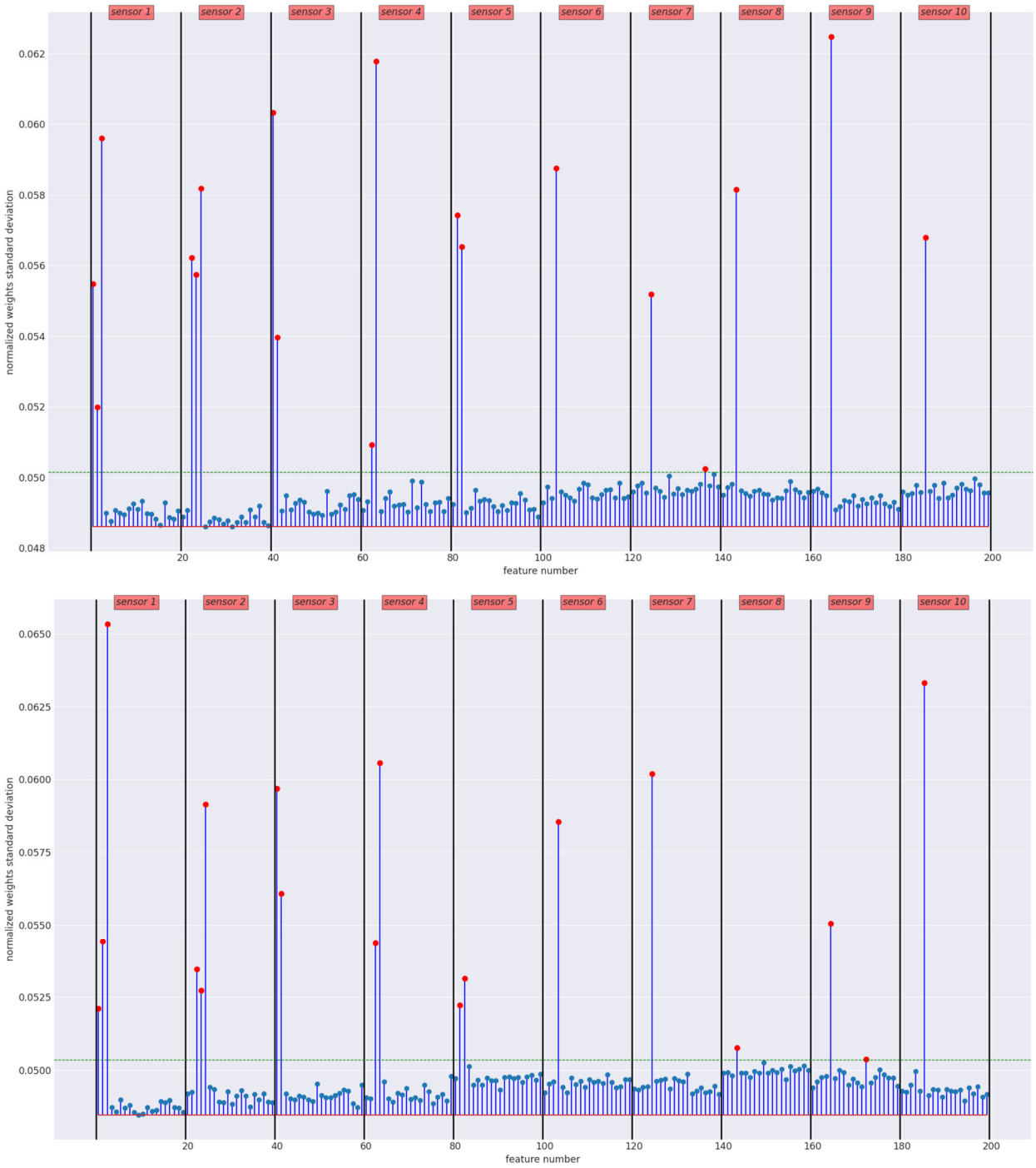

Figure 18: Selected features based on normalized weight standard deviations (heterogeneous feature uncertainty with $\sigma_{\phi_j} \sim U(0.1, 0.5)$ at two noise levels: $\epsilon \sim \mathcal{N}(0, 0.1)$ (top figure) and $\epsilon \sim \mathcal{N}(0, 0.5)$ (bottom figure)).

The feature selection model appears to be quite robust since it identifies the relevant features in the two test cases. However, the model selects extra features in both cases:

a. Homogeneous case and $\epsilon \sim \mathcal{N}(0,0.1)$, 2 features for sensor 7
b. Heterogeneous case and $\epsilon \sim \mathcal{N}(0,0.1)$, 2 features for sensor 7
c. Heterogeneous case and $\epsilon \sim \mathcal{N}(0,0.5)$, 2 features for sensor 9

These "errors" are due to the thresholding method, here the elbow method. However, the threshold is much harder to find especially for the heterogeneous test case where the noisy baseline (normalized weight standard deviations of irrelevant features) gets very close to the normalized weight standard deviations associated with relevant features.

5.8 NUMERICAL EXAMPLE: STRATH TESTBED

For the STRATH testbed, there are 99 sensor time series. Although the number of time series to be considered is reduced using methods of segmentation and redundancy evaluation as introduced before, there is still a larger number of time series remaining for analysis. To predict the CMM dimensions of the parts, another challenge is that the time series across different parts are very similar, especially for the first dataset of 81 parts in which there were no controlled variables. Consider the 'Force' signal as an example. It is considered as an important sensor signal in production. In Figure 19, the first dataset of 81 parts are plotted superimposed on each other. High similarity between the sensor signals implies that distinguishing features for classification are difficult to find.

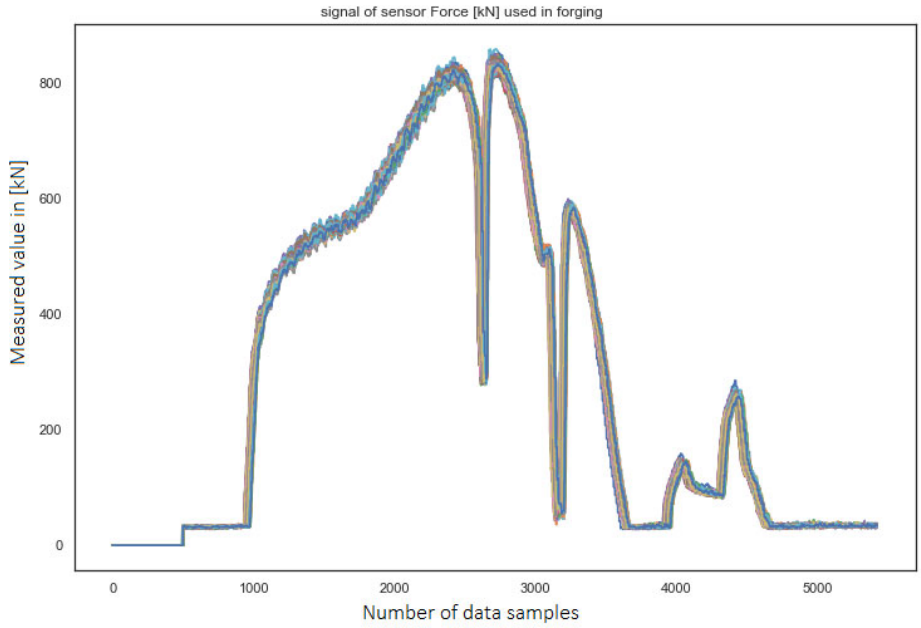

Figure 19: 'Force' signals plotted together for the first 81 parts.

5.8.1 Feature extraction

Three types of features have been extracted:

- Set 1: Statistics such as maximum, minimum, average, variance, kurtosis, etc., are extracted for each processing phase, as well as the difference between nominal and actual value of the time series such as position, speed, etc.

- Set 2: Features from the limited knowledge of the testbed [Haris 2019], such as the length of pick-up from induction coil to forging box, the temperature drop during the pick-up, etc.
- Set 3: Three-level wavelet transform has been applied to the data. Statistical features have been extracted from each frequency band. The reason for using the wavelet transform is its well-known time-frequency decomposition ability.

There are about 4000 features extracted from these three sets of features. The number of features from the wavelet transform alone is around 2700. Since in total there are only 177 parts for analysis, a potential problem would be heavy computational complexity and increased likelihood of overfitting.

5.8.2 Feature sifting

Consideration of the Pearson correlation between each feature and a measurand is used to reduce the number of statistical features from time domain for Set 1. Only those features with coefficient value that is higher than the threshold $\alpha = 0.5$ are retained. This operation typically reduces the number of features from over a thousand to a few hundred, depending on the CMM dimension. In Figure 20, for each sensor of interest, the correlation matrix between the measurands (CMM dimensions) and feature of maximum value during forging is shown as an illustration.

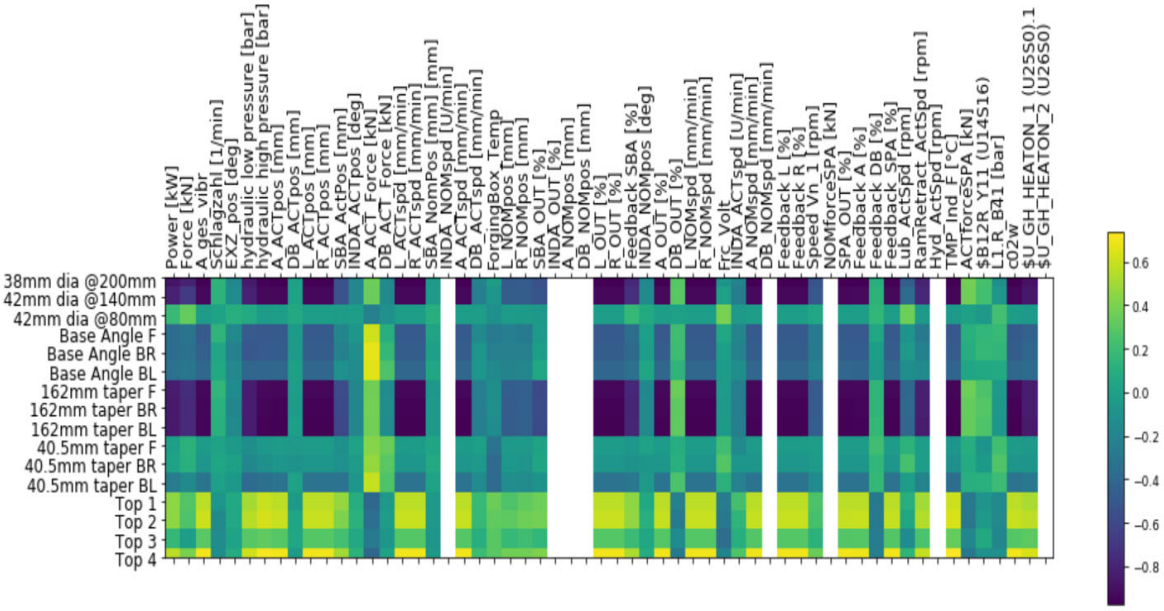

Figure 20: Correlation matrix between the measurands (CMM dimensions) and a feature (maximum value) extracted from forging phase.

After sifting, further analysis shows that many of the selected features are highly correlated with each other. For those with larger than $\beta = 0.85$ cross correlation, only one feature is retained. The objective is to reduce the linear dependence between features. The operation further reduces the number of features from hundreds to tens. In Table 1, the number of features selected after sifting is summarized. Interestingly, there are a few dimensions for which more features are selected than others, namely CMM dimension 0, 1, 2, 6, 7 and 8. These dimensions turn out to provide better prediction results.

CMM dimension	CMM dimension name	Number of selected features after sifting from Set1
0	38mm dia @200mm	23
1	42mm dia @140mm	20
2	42mm dia @80mm	2
3	Base Angle F	7
4	Base Angle BR	5
5	Base Angle BL	5
6	162mm taper F	22
7	162mm taper BR	25
8	162mm taper BL	24
9	40.5mm taper F	0
10	40.5mm taper BR	1
11	40.5mm taper BL	2
12	Top 1	5
13	Top 2	8
14	Top 3	1
15	Top 4	13

Table 1: The number of selected features in Set 1 after sifting.

Note that the values of hard thresholding, i.e., α and β, affect the number of selected features into the candidate pool, and therefore the computation complexity and training and the prediction accuracy. They may be chosen by empirical simulation. But their choice also depends on the training data size. For the STRATH testbed, the hard thresholding values are selected as the same value across the different dimensions for simplicity. Had the training data size been larger, it would be possible to reduce the threshold and allow more features into the candidate pool. It is also sensible to select different values for different CMM dimensions because some CMM dimensions are more difficult to predict than others. The threshold can be slightly lower to allow features with less strong correlation to be retained.

In Figure 21, for the CMM dimension “42mm dia @140mm”, the cross correlation of the selected features in Set 1 after sifting is shown. Twenty features are selected from Set 1.

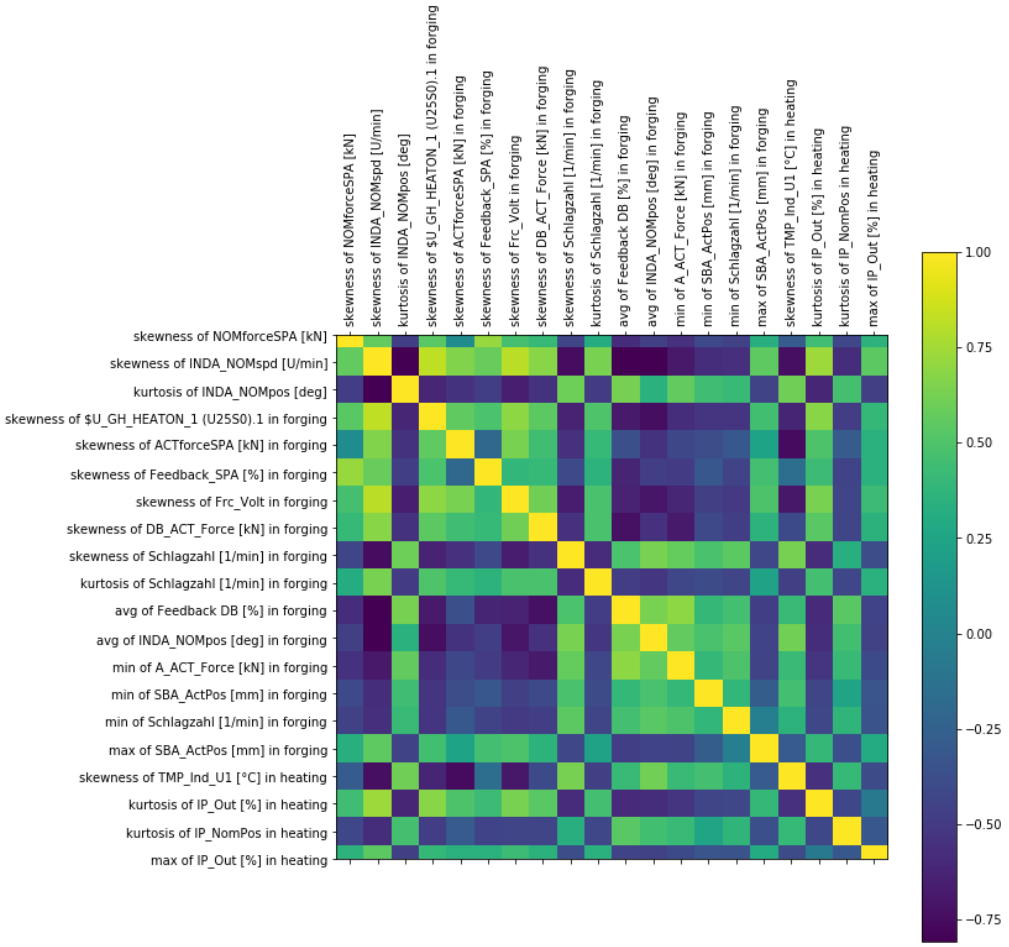

Figure 21: Correlation matrix between CMM dimension "42mm dia @140mm" and selected features from Set 1 after sifting.

For those features in Set 2, they are put directly into the candidate pool as they agree with the understanding of the system to a certain degree. For those features in Set 3 derived from the wavelet transform, the top *n* features are selected. Due to the small training data size, *n* = 10 is chosen for the STRATH testbed example. But again, if the training dataset were of larger size, the value *n* could be bigger to allow more candidates features.

After sifting, the feature selection algorithm with uncertainty awareness described in this section is applied. The STRATH testbed is an advanced forging machine with high accuracy and good reliability, meaning there is no uncertainty in the sensor data. However, in the whole project period, the three data batches came at different time. Data batch number 2, which contains experimental data for 50 parts, is quite different from the previous batch of 81 parts as they are controlled experiments instead of produced by the normal machine forging operation. Data batch number 3 of 46 parts, is like data batch number 2 in the sense that the controlled variables are set within the same ranges. In terms of features, their values change every time a new data batch arrives, which is another type of uncertainty. In other words, the size of dataset and different system setting of the testbed leads to uncertainty in the features, although not in the conventional sense.

With the RGS feature selection algorithm, given the same set of features (i.e., the selected features after sifting in Set 1 together with the features in Set 2 and the top 10 features in Set 3), different features are selected for different CMM dimensions. Encouragingly, for the two

CMM dimensions, “42mm dia @80mm” and “42mm dia @140mm”, the desired feature is selected, as shown in Table 2 and Table 3, i.e., the ‘First Peak’ in the ‘Force’ signal, shown by the red dot in Figure 7.

SelectedFeaturesNames	SelectedFeaturesScores
Wavelet Feature 1	0.024938055
Wavelet Feature 2	0.024326445
Transfer Time	0.022323269
Temperature Drop in Pickup	0.022103949
Forging Phase 5 Force Integral [kNs]	0.021898036
Wavelet Feature 3	0.021119532
Wavelet Feature 0	0.020972517
Wavelet Feature 4	0.020404978
First Peak	0.019872225
Wavelet Feature 5	0.018847128
Forging Box Temperature Start	0.018709004
Forging Box Temperature End	0.018703022
Forging Temperature ROI	0.018457053
Wavelet Feature 7	0.01828926
Forging Time	0.017972573
Wavelet Feature 9	0.017785316
Forging Temperature Start	0.017425446
Wavelet Feature 8	0.017085748

Table 2: Selected features for “42mm dia @80mm”.

SelectedFeaturesNames	SelectedFeaturesScores
Temperature Drop in Pickup	0.032477528
First Peak	0.031776023
Forging Box Temperature Start	0.030670333
Forging Box Temperature End	0.030661767
Transfer Time	0.029294697
Forging Phase 5 Force Integral [kNs]	0.029085664
Forging Temperature ROI	0.029070997
Forging Time	0.02753277
Forging Temperature Start	0.025492693
Forging Phase 1 Force Integral [kNs]	0.024250813
Forging Phase 4 Force Max [kN]	0.024169876

Table 3: Selected features for “42mm dia @140mm”.

For the above two CMM dimensions, it appears that the features selected are largely from Set 2. But, for the dimension such as “38mm dia @200mm”, 10 out 18 features chosen are from wavelet features, meaning all 10 wavelet features have been selected. Therefore, for STRATH testbed, there is no concrete conclusion that features selected from limited knowledge of the testbed are better than those selected by a black box approach. In fact, for those CMM dimensions with good prediction results, a high proportion of features come from the wavelet analysis.

5.8.3 Regression with selected features

With the selected features, linear regression is carried out for the CMM dimension prediction task. For the CMM dimension "42mm dia @140mm" the regression result is shown below in Figure 22. For the regression, the test data is 20 % of the dataset. For a total of 177 parts, 22 parts are used as test data while the other are used for training. The prediction results are closely matched with the measured values with R^2 =0.9 in this case.

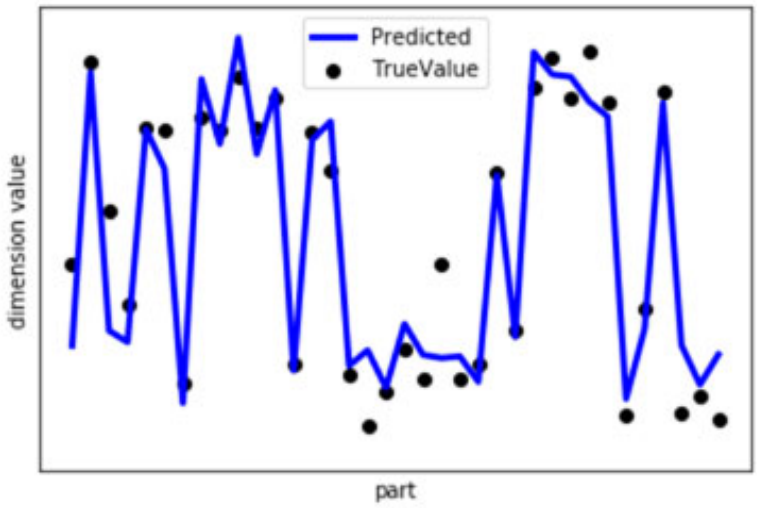

Figure 22: Linear regression results for CMM dimension "42mm dia @140mm".

Using *k* folds validation, and *k* = 8, for other major CMM dimensions, the prediction results are summarized in Table 4.

CMM dimension	CMM dimension name	R^2
0	38mm dia @200mm	0.936
1	42mm dia @140mm	0.902
2	42mm dia @80mm	0.525
3	Base Angle F	0.574
4	Base Angle BR	0.572
5	Base Angle BL	0.530
6	162mm taper F	0.884
7	162mm taper BR	0.910
8	162mm taper BL	0.940
9	40.5mm taper F	0.255
10	40.5mm taper BR	0.295
11	40.5mm taper BL	0.342

Table 4: Cross validation of prediction results for different CMM dimensions.

The prediction accuracy varies with CMM dimension. For some of the CMM dimensions, results are far from ideal. But considering the moderate data size available for training, the complexity of forging processing and the close range of measurands, several CMM dimensions can provide good prediction results. Noticing that there is no fine tuning in the parameter selection for thresholding as well as other stages, further improvement would be expected.

6 TIMING AND SYNCHRONISATION EFFECTS IN INDUSTRIAL SENSOR NETWORKS

6.1 BACKGROUND

Many measurements important in manufacturing relate to dynamical systems evolving over time. Sensors may individually have access to an accurate clock but for multiple, spatially distributed sensors, it is essential that the sensors are working to a common timescale, for example, through traceability to UTC, so that the data they record can be analysed correctly. Accurate timestamping can be an important diagnostic aid to establishing, for example, which cause-effect relationships are feasible, and which are infeasible, or providing information about the path of a signal (such as vibrations) traversing a structure.

In addition to ensuring that the data recorded by different sensors are synchronized, such data can be subject to timing errors, such as jitter. Pre-processing of the sensor signals can be applied to evaluate the uncertainties of the effects of noise and jitter and to correct for such effects. It is important to understand the uncertainty of the resulting pre-processed signals, and to be able to propagate those uncertainties through any subsequent processing or aggregation of the sensor signals.

The paper [White, 2017] describes a number of well-known algorithms for interpolating tabulated data, including straight-line, spline, Lagrange, and least-squares interpolation, and provides a comparison of those algorithms with respect to the propagation of uncertainty, interpolation error and sensitivity to data spacing. Although it is recognised that there is no interpolating equation that is best for all applications, the paper concludes that an interpolation scheme in which a polynomial is fitted by least-squares is good in the sense of having a low propagated uncertainty, continuity to the chosen order of the polynomial, and immunity to large gaps in the data. That work is extended here in ways that are important for the application of industrial sensor networks, which include (a) treating uncertainty in the values of the independent variable, i.e., the timestamps for the data recorded by an individual sensor, arising from jitter, (b) treating uncertainty in the time origins for the data recorded by different sensors arising from synchronisation errors, and (c) providing covariances between pairs of signal values that are corrected for the effects of jitter and synchronisation errors. In line with [White, 2017], an interpolating scheme based on using a low order polynomial function to provide a local approximation to the data is used.

6.2 MODEL OF NOISE AND JITTER

Let $v = s(t)$ describe the underlying time-dependent signal that is measured by the sensor, and assume that s is twice continuously differentiable with the first and second derivative functions $s^{(1)}(t)$ and $s^{(2)}(t)$ not identically zero. Let (t_i, v_i) be the measured data recorded by the sensor with the time values t_i usually uniformly spaced. A model for the measured data is

$$v_i = s(t_i + \delta t_i) + \delta v_i,$$

where δt_i is a timing error representing signal jitter and δv_i is an error representing signal noise. The model says that the recorded signal value differs from the 'true' value of the signal at a 'true' value of time that, in turn, differs from the recorded value of time. The errors δt_i and δv_i are assumed to be samples drawn independently from probability distributions described, respectively, by probability density functions $g_J(\xi)$ and $g_N(\eta)$ that are independent of time and of the signal and of each other. The probability distributions have zero expectations, and their variances are denoted, respectively, by τ^2 and ω^2, which describe the levels of jitter and noise in the data recorded by the sensor. Regarded as a continuous function of time, the above model takes the form

$$v(t) = s(t + \xi) + \eta.$$

It is shown [Cox, 1993] that

$$E[v(t)] = s(t) + \frac{1}{2}\tau^2 s^{(2)}(t)$$

and, approximately,

$$\mathrm{V}[v(t)] = \tau^2\{s^{(1)}(t)\}^2 + \omega^2.$$

The first result says that the expected value of $v(t)$ differs from $s(t)$ by the amount $\tau^2 s^{(2)}(t)/2$, which represents a time-dependent bias or systematic error. The bias is independent of the signal noise and greatest when the magnitude of the second derivative of the underlying signal is greatest. When $\tau^2 = 0$, i.e., there is no jitter, the measured signal is unbiased, as would be expected. From the second result, it is seen that when $s^{(1)}(t)$ is zero, the jitter makes no contribution to the variance of the measured signal, and the variance is greatest when the magnitude of the first derivative of the underlying signal is greatest.

6.3 ALGORITHM FOR THE REMOVAL OF NOISE AND JITTER

It follows from the above expressions that an unbiased estimate $p(t)$ of $s(t)$ can be constructed by approximating the measured signal $v(t)$ by $p(t) + \tau^2 p^{(2)}(t)/2$. The approximation is determined using a squared weighting function inversely proportional to $\tau^2\{p^{(1)}(t)\}^2 + \omega^2$. It is noted that if $s(t)$ is a cubic polynomial function and the probability distribution for the jitter is symmetric then $\mathrm{E}[v(t)] = s(t) + \tau^2 s^{(2)}(t)/2$ holds exactly. It follows that the only essential limitation of the proposed approach is the ability for the underlying signal to be adequately represented by a cubic polynomial. In practice, this limitation is minimized by choosing to approximate the measured signal over a sequence of possibly short intervals of time.

Let $(t_i, v_i), i = 1, 2, \ldots,$ be the measured data recorded by the sensor and let the variances of jitter and noise be denoted, respectively, by τ^2 and ω^2. Choose an integer n such that $N = 2n + 1 > 4$ and a tolerance $\varepsilon > 0$. To obtain an estimate $\hat{s}_k$ of $s(t_k)$ at time t_k for $k > n$, the following steps are applied in terms of the data $(t_j, v_j), j = k - n, \ldots, k + n$:

Step 1: Compute a least-squares cubic polynomial fit to the data $(t_j, v_j), j = k - n, \ldots, k + n,$ to give an initial approximation $p_{k,0}(t)$ to $s(t)$ for times t between t_{k-n} and t_{k+n}.
Step 2: Evaluate $p_{k,0}^{(1)}(t)$ at each time $t_j, j = k - n, \ldots, k + n.$
Step 3: Compute a new approximation $p_k(t) = a_{k,0} + a_{k,1}t + a_{k,2}t^2 + a_{k,3}t^3$ to $s(t)$ by fitting the data $(t_j, v_j), j = k - n, \ldots, k + n,$ in a weighted least-squares sense by the function

$$p_k(t) + \frac{1}{2}\tau^2 p_k^{(2)}(t) \equiv a_{k,0} + a_{k,1}t + a_{k,2}(t^2 + \tau^2) + a_{k,3}(t^3 + 3\tau^2 t)$$

using weights

$$w_j = \sqrt{\frac{1}{\tau^2\{p_{k,0}^{(1)}(t_j)\}^2 + \omega^2}}.$$

Step 4: Set $p_{k,0}(t) = p_k(t)$ and repeat from step 2 until the process has stabilized, for example, when $\max_j |p_k(t_j) - p_{k,0}(t_j)| < \varepsilon$.
Step 5: Evaluate $\hat{s}_k = p_k(t_k)$.

The above steps are applied to data within the sequence of time windows $[t_{k-n}, t_{k+n}], k = n + 1, n + 2, \ldots,$ to obtain the sequence of estimates $\hat{s}_k, k = n + 1, n + 2, \ldots,$ of the underlying signal. The algorithm is applied in the spirit of a "moving average", only requiring data to be available in the neighborhood of the time t_k at which an estimate is to be determined. In this way, the algorithm can run continuously and applied to data as it is recorded by a sensor, although there

is a time lag of n time-steps between the time t_{k+n} at which the (last) measured signal value is recorded and the time t_k at which an estimated signal value can be evaluated.

6.4 UNCERTAINTY EVALUATION FOR THE SIGNAL ESTIMATES

For time $t = t_k$, let $\hat{s}_k$ be the estimate of $s(t_k)$ provided by the algorithm described above. Formally, the estimate $\hat{s}_k$ is related to the measured signal values $\boldsymbol{v}_k$ through the linear measurement model

$$\hat{s}_k = \boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{v}_k = \boldsymbol{t}_k^{\mathrm{T}}\widetilde{\boldsymbol{X}}_k^{+}\boldsymbol{W}_k\boldsymbol{v}_k,$$

where $\boldsymbol{t}_k = (1, t_k, t_k^2, t_k^3)^{\mathrm{T}}$ and $\widetilde{\boldsymbol{X}}_k^{+}$ is the pseudoinverse of the matrix $\widetilde{\boldsymbol{X}}_k = \boldsymbol{W}_k\boldsymbol{X}_k$ in which $\boldsymbol{X}_k$ denotes the $N \times 4$ matrix whose rows contain the elements $1, t_j, (t_j^2 + \tau^2), (t_j^3 + 3\tau^2 t_j)$ and $\boldsymbol{W}_k$ is the $N \times N$ diagonal matrix containing the weights w_j.

By applying the "Law of Propagation of Uncertainty" (LPU) of the GUM [GUM, 2008], if follows that the variance $V[\hat{s}_k] = u^2(\hat{s}_k)$ of the estimate $\hat{s}_k$ is given by

$$\mathrm{V}[\hat{s}_k] = \boldsymbol{c}_k^{\mathrm{T}}\mathrm{V}[\boldsymbol{v}_k]\boldsymbol{c}_k,$$

where $\mathrm{V}[\boldsymbol{v}_k]$ is the covariance matrix of $\boldsymbol{v}_k$, a diagonal matrix whose diagonal elements are the variances of $v_j, j = k - n, \ldots, k + n$. Since

$$\mathrm{V}[v_j] = \tau^2\{s^{(1)}(t_j)\}^2 + \omega^2,$$

these variances are not known because they depend on the unknown underlying sensor signal, and so it is necessary to decide how to approximate them. We choose to use the approximation

$$\mathrm{V}[v_j] = \tau^2\{p_j^{(1)}(t_j)\}^2 + \omega^2,$$

where $p_j(t)$ is the approximation to $s(t)$ constructed when forming the approximation $\hat{s}_j$ to $s(t_j)$. This choice of approximation means that the evaluation of the variance $\mathrm{V}[\hat{s}_k]$ depends on the analysis of the data within the time window $[t_{k-n}, t_{k-n}]$ as well as the analysis of the data within neighboring time windows.

Now consider a second time $t = t_l$ with $l > k$. If $l - n > k + n$, then the vectors $\boldsymbol{v}_k = (v_{k-n}, \ldots, v_{k+n})^{\mathrm{T}}$ and $\boldsymbol{v}_l = (v_{l-n}, \ldots, v_{l+n})^{\mathrm{T}}$ have no elements in common and the estimates $\hat{s}_k$ and $\hat{s}_l$ are uncorrelated. In the case that the vectors do have elements in common, let $\boldsymbol{v}_{kl}$ be the vector containing without repetition, and in order, the measured signal values on which $\hat{s}_k$ and $\hat{s}_l$ depend, i.e., $\boldsymbol{v}_{kl} = (v_{k-n}, \ldots, v_{l-n}, \ldots, v_{k+n}, \ldots, v_{l+n})^{\mathrm{T}}$.The estimate $\hat{s}_k$ is related to the measured signal values $\boldsymbol{v}_{kl}$ through the linear measurement model

$$\hat{s}_k = \boldsymbol{c}_k^{\mathrm{T}}\boldsymbol{v}_k = \boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{v}_{kl},$$

where $\boldsymbol{d}_k^{\mathrm{T}}$ is constructed from $\boldsymbol{c}_k^{\mathrm{T}}$ by including zeros at the end of the vector that multiply the terms $v_{k+n+1}, \ldots, v_{l+n}$. Similarly, $\hat{s}_l$ is related to the measured signal values $\boldsymbol{v}_{kl}$ through the linear measurement model

$$\hat{s}_l = \boldsymbol{c}_l^{\mathrm{T}}\boldsymbol{v}_l = \boldsymbol{d}_l^{\mathrm{T}}\boldsymbol{v}_{kl},$$

where $\boldsymbol{d}_l^{\mathrm{T}}$ is constructed from $\boldsymbol{c}_l^{\mathrm{T}}$ by including zeros at the start of the vector that multiply the terms $v_{k-n}, \ldots, v_{l-n-1}$. It follows that

$$\begin{pmatrix}\hat{s}_k\\ \hat{s}_l\end{pmatrix} = \begin{pmatrix}\boldsymbol{c}_k^{\mathrm{T}}\\ \boldsymbol{c}_l^{\mathrm{T}}\end{pmatrix}\boldsymbol{v}_{kl} = \boldsymbol{C}_{kl}\boldsymbol{v}_{kl},$$

and, again applying the GUM's LPU, the covariance matrix for the estimates $\hat{s}_k$ and $\hat{s}_l$ is

$$\mathrm{V}\left[\begin{pmatrix}\hat{s}_k\\ \hat{s}_l\end{pmatrix}\right] = \boldsymbol{C}_{kl}\mathrm{V}[\boldsymbol{v}_{kl}]\boldsymbol{C}_{kl}^{\mathrm{T}},$$

a 2×2 matrix containing $\mathrm{V}[\hat{s}_k] = u^2(\hat{s}_k)$ and $\mathrm{V}[\hat{s}_l] = u^2(\hat{s}_l)$ on its diagonal and the covariance $\mathrm{cov}[\hat{s}_k, \hat{s}_l]$ on its off-diagonal.

6.5 ACCOUNTING FOR A KNOWN SYNCHRONIZATION ERROR

Suppose two sensors record, respectively, the data $(t_{1,i}, v_{1,i})$ and $\left(t_{2,i}, v_{2,i}\right), i = 1,2, \ldots$, for which there is a *known* synchronization error δt_0, i.e.,

$$\delta t_0 = t_{1,i} - t_{2,i}, \qquad i = 1, 2, \ldots$$

where δt_0 is assumed to be small compared to the uniform sampling intervals $(t_{1,i+1} - t_{1,i})$ and $(t_{2,i+1} - t_{2,i})$, respectively, for the two sensors. Let τ_1^2 and ω_1^2 denote the variances of jitter and noise for the first sensor and, similarly, τ_2^2 and ω_2^2 the variances for the second sensor. The treatment described above is used to provide estimates $\hat{s}_{1,k}$ at the time-points $t = t_{1,k}$ of the signal measured by the first sensor with associated uncertainty information. To provide estimates $\hat{s}_{2,k}$ at the *same* time-points $t = t_{1,k}$ of the signal measured by the second sensor with associated uncertainty information, it is enough to apply that treatment except that the cubic polynomial functions $p_{2,k}(t)$ determined from the sequences of data $\left(t_{2,j}, v_{2,j}\right), j = k - n, \ldots, k + n$, are evaluated at $t = t_{1,k}$ rather than at $t = t_{2,k}$. In other words, **step 5** is replaced by

"**Step 5**: Evaluate $\hat{s}_{2,k} = p_{2,k}\left(t_{1,k}\right)$. "

and, in the evaluation of uncertainties, the vector $\boldsymbol{t}_k$, which is denoted by $\boldsymbol{t}_{2,k}$ for the second sensor, is calculated as $\boldsymbol{t}_{2,k} = (1, t_{1,k}, t_{1,k}^2, t_{1,k}^3)^{\mathrm{T}}$. In this way, a correction for the known synchronization error is applied to the signal estimates determined for the second sensor and uncertainties associated with the corrected estimates are evaluated.

6.6 ACCOUNTING FOR AN UNKNOWN SYNCHRONIZATION ERROR

As before, let (t_i, v_i) be the measured data recorded by the sensor with the time values t_i usually uniformly spaced. A model for the measured data is

$$v_i = s(t_i + \delta t_0 + \delta t_i) + \delta v_i,$$

where δt_0 is a fixed but *unknown* synchronization error. The errors δt_0, δt_i and δv_i are assumed to be samples drawn independently from probability distributions described, respectively, by probability density functions $g_0(\xi_0)$, $g_J(\xi)$ and $g_N(\eta)$ that are independent of time and of the signal and of each other. The probability distributions have zero expectations, and their variances are denoted, respectively, by τ_0^2, τ^2 and ω^2, which describe the levels of synchronization error, jitter and noise in the data recorded by the sensor. Regarded as a continuous function of time, the above model takes the form

$$v(t) = s(t + \xi_0 + \xi) + \eta.$$

Following the same analysis as given before, the expectation $\mathrm{E}[v(t)]$ of the measured signal $v(t)$ at time t is

$$\mathrm{E}[v(t)] = s(t) + \frac{1}{2}(\tau_0^2 + \tau^2)s^{(2)}(t),$$

and the variance $\mathrm{V}[v(t)]$ is approximately

$$\mathrm{V}[v(t)] = (\tau_0^2 + \tau^2)\{s^{(1)}(t)\}^2 + \omega^2.$$

Because of the presence of the common synchronization error, the measured signal $v(t)$ and $v(t')$ at time-points t and t' that are distinct are correlated with a covariance that is approximately

$$\mathrm{cov}[v(t), v(t')] = \tau_0^2 s^{(1)}(t)s^{(1)}(t').$$

Here, it is seen that when either $s^{(1)}(t)$ or $s^{(1)}(t')$ is zero, the covariance of the measured signal at the two times is zero, and the covariance is greatest when the magnitude of the product of the first derivatives of the underlying signal at those times is greatest.

The expressions for $\mathrm{E}[v(t)]$, $\mathrm{V}[v(t)]$ and $\mathrm{cov}[v(t), v(t')]$ form the basis of an extension to the algorithm for removing noise and jitter from data representing the measured signal that accounts for a fixed and unknown synchronization error. Since an estimate of the value of that error is zero, no correction is made for it, but the estimates $\hat{s}_k$ of the signal values $s(t_k)$ depend on the uncertainty of the correction and that uncertainty is also propagated to the uncertainties of the estimates. Specifically, step 3 is replaced by

"**Step 3**: Compute a new approximation $p_k(t) = a_{k,0} + a_{k,1}t + a_{k,2}t^2 + a_{k,3}t^3$ to $s(t)$ by fitting the data $(t_j, v_j), j = k - n, \ldots, k + n,$ in a generalized least-squares sense by the function

$$p_k(t) + \frac{1}{2}(\tau_0^2 + \tau^2)p_k^{(2)}(t) \equiv a_{k,0} + a_{k,1}t + a_{k,2}(t^2 + \tau_0^2 + \tau^2) + a_{k,3}(t^3 + 3\{\tau_0^2 + \tau^2\}t)$$

using covariance matrix $\boldsymbol{V}$ with diagonal elements

$$V_{j,j} = (\tau_0^2 + \tau^2)\left\{p_{k,0}^{(1)}(t_j)\right\}^2 + \omega^2$$

and off-diagonal elements

$$V_{j,l} = \tau_0^2 p_{k,0}^{(1)}(t_j)p_{k,0}^{(1)}(t_l).$$ "

The variance $V[\hat{s}_k] = u^2(\hat{s}_k)$ of the estimate $\hat{s}_k = p_k(t_k)$ is given by

$$\mathrm{V}[\hat{s}_k] = \boldsymbol{c}_k^{\mathrm{T}}\mathrm{V}[\boldsymbol{v}_k]\boldsymbol{c}_k$$

in which $\boldsymbol{c}_k$ is replaced by

$$\boldsymbol{c}_k^{\mathrm{T}} = \boldsymbol{t}_k^{\mathrm{T}}\widetilde{\boldsymbol{X}}_k^{+}\boldsymbol{L}_k^{\mathrm{T}}, \qquad \widetilde{\boldsymbol{X}}_k = \boldsymbol{L}_k^{\mathrm{T}}\boldsymbol{X}_k,$$

where $\boldsymbol{V}^{-1} = \boldsymbol{L}_k\boldsymbol{L}_k^{\mathrm{T}}$ is the Cholesky factorization of $\boldsymbol{V}^{-1}$, and $\mathrm{V}[\boldsymbol{v}_k]$ is replaced by a matrix with diagonal elements

$$(\tau_0^2 + \tau^2)\left\{p_j^{(1)}(t_j)\right\}^2 + \omega^2$$

and off-diagonal elements

$$\tau_0^2 p_j^{(1)}(t_j)p_l^{(1)}(t_l).$$

The covariance matrix for the estimates $\hat{v}_k$ and $\hat{v}_l$ at two distinct time-points is

$$\mathrm{V}\left[\begin{pmatrix}\hat{s}_k\\ \hat{s}_l\end{pmatrix}\right] = \boldsymbol{C}_{kl}\mathrm{V}[\boldsymbol{v}_{kl}]\boldsymbol{C}_{kl}^{\mathrm{T}},$$

where, as before, $\boldsymbol{v}_{kl}$ is the vector containing without repetition, and in order, the measured signal values on which $\hat{s}_k$ and $\hat{s}_l$ depend. However, it is no longer the case that the covariance of the estimates is zero when the time-points are sufficiently far apart, because a component v_k of $\boldsymbol{v}_k$ can be expected to be correlated with a component v_l of $\boldsymbol{v}_l$ as they are both influenced by the synchronization effect. Note that in this procedure there is no attempt to estimate the synchronization error δt_0, or to correct for it, but the procedure is expected to give on average over all synchronization problems a reasonable reconstruction of the signal.

6.7 ESTIMATING THE JITTER AND NOISE VARIANCES

In the above sections it is assumed that τ^2 and ω^2, the levels of jitter and noise in the data, are known. In this section, the problem of estimating those levels is considered, and a Bayesian approach is described for that purpose.

As before, let $(t_j, v_j), j = k-n, \ldots, k+n$, denote data within a time window of length $N = 2n+1$ centered on time t_k. In the case that there is no synchronization error, the probability of observing the data conditioned on values $\boldsymbol{a}$ of the parameters of the model $p(t;\boldsymbol{a})$ for the underlying signal and on values τ^2 and ω^2 of the variances of the distributions for, respectively, jitter and noise, is

$$h(\boldsymbol{v}|\boldsymbol{a},\tau^2,\omega^2) = \prod_{j=k-n}^{k+n} h\left(v_j\middle|\boldsymbol{a},\tau^2,\omega^2\right),$$

where

$$h\left(v_j\middle|\boldsymbol{a},\tau^2,\omega^2\right) \propto \frac{1}{\sqrt{\tau^2\{p^{(1)}(t_j;\boldsymbol{a})\}^2+\omega^2}}\exp\left\{-\frac{\left(v_j - p\left(t_j;\boldsymbol{a}\right) - \frac{1}{2}\tau^2 p^{(2)}\left(t_j;\boldsymbol{a}\right)\right)^2}{2\left(\tau^2\left\{p^{(1)}\left(t_j;\boldsymbol{a}\right)\right\}^2+\omega^2\right)}\right\}$$

Consider prior knowledge for $\boldsymbol{a}$ captured by an uninformative, uniform distribution, viz., $g(\boldsymbol{a}) \propto 1$. The prior distribution chosen for τ^2 (and similarly for ω^2) is the inverse-gamma distribution having probability density function

$$g(\tau^2) \propto (\tau^2)^{-(\alpha+1)}\exp\left(-\frac{\beta}{\tau^2}\right),$$

with shape parameter $\alpha = m_0/2$ and scale parameter $\beta = m_0\tau_0^2/2$. Here, τ_0^2 represents a prior estimate of τ^2 and m_0 encodes the "degree of belief" associated with the prior estimate and can be considered to relate to the amount of data used to obtain the estimate. The inverse-gamma distribution is used because it is the conjugate prior distribution for the variance parameter in a normal linear regression model. A large value of m_0 implies that the prior knowledge about τ^2 is informative and a small value that it is uninformative. The limiting cases $m_0 \to \infty$ and $m_0 \to 0$ correspond to when τ^2 is known (and equal to τ_0^2) and τ^2 is unknown (and to be estimated from the data), respectively. These two extreme cases can be treated in an ordinary least-squares analysis. However, a Bayesian analysis is more subtle in the sense that it permits information about τ^2 expressed by a prior distribution to be combined with information about τ^2 contained in the data.

Applying Bayes' theorem, the posterior distribution for $\boldsymbol{a}$, τ^2 and ω^2 is

$$g(\boldsymbol{a}, \tau^2, \omega^2|\boldsymbol{v}) \propto h(\boldsymbol{v}|\boldsymbol{a}, \tau^2, \omega^2) g(\boldsymbol{a}) g(\tau^2) g(\omega^2).$$

A Markov-Chain Monte-Carlo method, based on a random-walk Metropolis-Hastings algorithm, can be used to obtain samples from the posterior distribution in terms of which estimates of $\boldsymbol{a}$, τ^2 and ω^2 and uncertainties associated with the estimates are evaluated. The proposal distribution for the algorithm is taken to be a joint Gaussian distribution whose expectation is given by the values of the parameters $\boldsymbol{a}$, τ^2 and ω^2 that maximize the (logarithm of) the posterior probability density and whose covariance matrix is given by the inverse of the Hessian of the posterior probability density at those values.

The calculation is more computationally expensive than that for the algorithm in which τ^2 and ω^2 are assumed to be known and, consequently, it is only intended to be applied occasionally. For example, the calculation might be used to provide values for τ^2 and ω^2 to be used to "initialize" the algorithm in which τ^2 and ω^2 are assumed to be known. Furthermore, the calculation might be applied periodically to check whether the estimates of τ^2 and ω^2 have changed and the algorithm needs to be "re-initialized".

6.8 SIMULATED EXAMPLE

The methods described in this chapter are illustrated using the simulated signal

$$s(t) = ae^{-bt}\sin(2\pi f t), \qquad 0 \le t \le 1,$$

with $a = 10$, $b = 2$ and $f = 2$. The signal is sampled with a uniform timing spacing of 0.01 s corresponding to a sampling frequency of 100 Hz, and the jitter and noise levels are set as $\tau = 0.002$ and $\omega = 0.005$, respectively. The data pre-processing algorithm described in sections 6.3 and 6.4, and Bayesian analysis of section 6.7, are applied to data to time windows of length $N = 15$ (corresponding to $n = 7$).
In the first case, prior distributions for τ^2 and ω^2 are defined by estimates equal to the (true) values of the variances used in the simulation and degrees of freedom m_0 of 50. It follows that the prior distributions are more informative about τ^2 and ω^2 than are the measured signal values contained in the time window. Figures 23 and 24 compare the prior distributions for τ^2 and ω^2 in the form of inverse-gamma distributions with the posterior distributions provided by a Markov-Chain Monte-Carlo sampling algorithm using 100 chains each of length 10,000 and a burn-in of 1,000 samples. It is seen that the combination of the prior information and the data produces a posterior distribution that is slightly less dispersed than the prior distribution. Table 5 lists, for five different non-overlapping time windows, the 0.025–, 0.500– and 0.975–percentiles of the posterior distributions for τ and ω, respectively.

Figure 25 shows the distribution for the estimate of the signal value at the middle time-point in the time window obtained from the Bayesian analysis (shown as a scaled frequency distribution) and the Gaussian distributions obtained using the described data pre-processing algorithm with τ and ω replaced by their true values (red curve) and by the medians of the posterior distributions (orange curve). It is seen that there is good agreement between the distributions indicating that the data pre-processing algorithm, which evaluates uncertainties by applying the GUM procedure, provides reliable information about the quality of the obtained estimates of the signal.

time window	jitter τ			noise ω		
1	0.0016	0.0019	0.0023	0.0041	0.0049	0.0061
2	0.0017	0.0020	0.0024	0.0042	0.0050	0.0061
3	0.0017	0.0020	0.0024	0.0041	0.0050	0.0061
4	0.0016	0.0019	0.0023	0.0041	0.0049	0.0061
5	0.0016	0.0019	0.0023	0.0041	0.0049	0.0060

Table 5: For five different non-overlapping time windows, the 0.025–, 0.500– and 0.975–percentiles of the posterior distributions for τ and ω. The prior distributions have degrees of freedom $m_0 = 50$.

time window	jitter τ			noise ω		
1	0.0017	0.0023	0.0033	0.0028	0.0044	0.0087
2	0.0014	0.0018	0.0025	0.0029	0.0045	0.0083
3	0.0012	0.0016	0.0023	0.0030	0.0043	0.0064
4	0.0017	0.0023	0.0033	0.0028	0.0045	0.0088
5	0.0014	0.0018	0.0024	0.0032	0.0043	0.0061

Table 6: For five different non-overlapping time windows, the 0.025–, 0.500– and 0.975–percentiles of the posterior distributions for τ and ω. The prior distributions have degrees of freedom $m_0 = 5$.

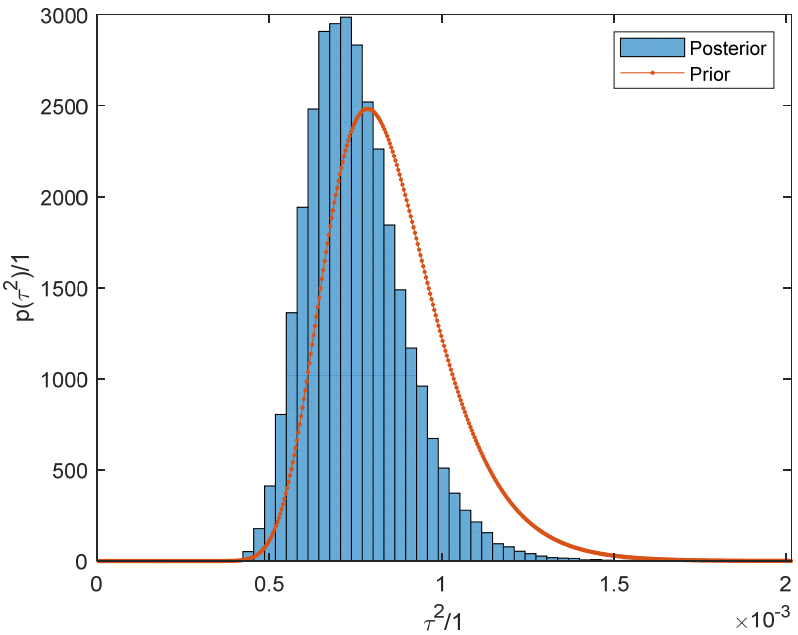

Figure 23: The prior and posterior distributions for τ^2 from the Bayesian analysis of data within a time window of length $N = 15$ and prior distributions defined by $m_0 = 50$.

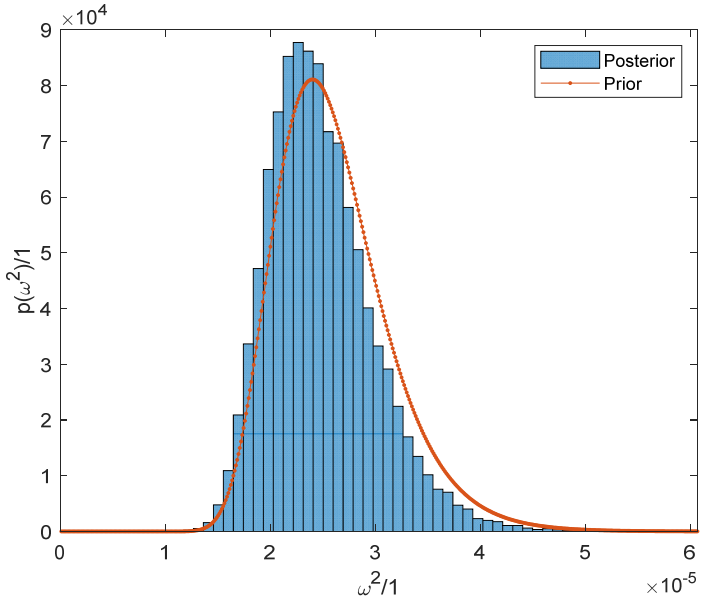

Figure 24: The prior and posterior distributions for ω^2 from the Bayesian analysis of data within a time window of length $N = 15$ and prior distributions defined by $m_0 = 50$.

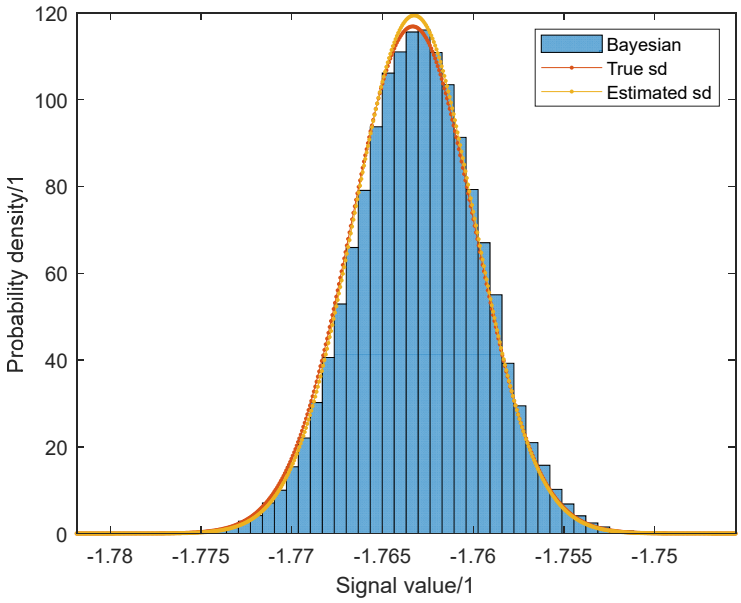

Figure 25: Distribution of the signal estimate obtained from a Bayesian analysis of data within a time window of length $N = 15$ and prior distributions defined by $m_0 = 50$ (histogram), and the Gaussian distributions obtained using the data pre-processing algorithm with the true values of the jitter and noise variances (red) and the medians of the posterior distributions (orange).

In the second case, prior distributions for τ^2 and ω^2 are defined by estimates equal to the (true) values of the variances used in the simulation and degrees of freedom m_0 of 5. It follows that the prior distributions are less informative about τ^2 and ω^2 than are the measured signal values contained in the time window, and this is reflected in Figures 26 and 27 that compare

the prior and posterior distributions for the two variances. The results listed in Table 6 show that the standard deviations τ and ω are less well determined when the prior information is less informative. Finally, Figure 28 compares the distributions for the estimate of the signal value at the middle time-point in the time window, which shows that, perhaps surprisingly, there is reasonable agreement between those distributions. The results suggest that the data pre-processing algorithm is not too sensitive to the values of the standard deviations τ and ω used in that algorithm, and the algorithm provides reliable information about the quality of the obtained estimates of the signal.

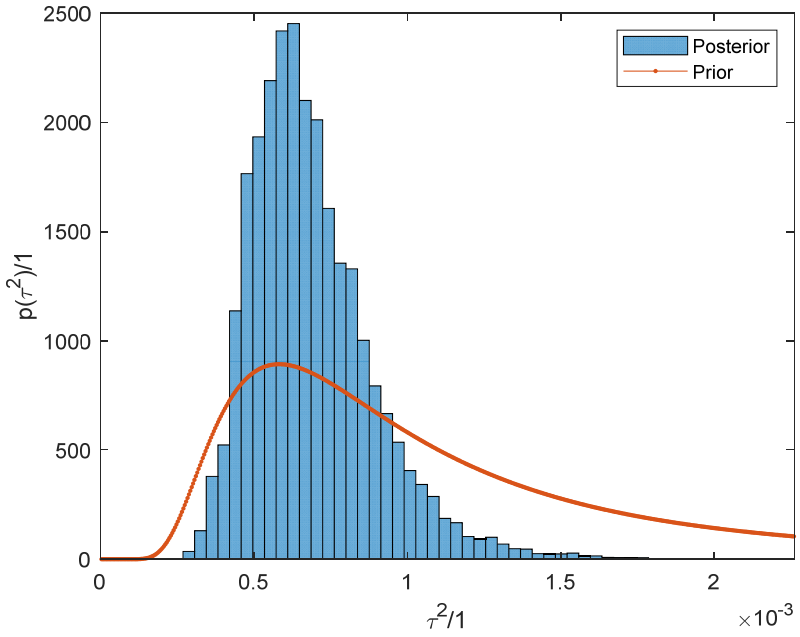

Figure 26: As Figure 23, except the prior distributions are defined by $m_0 = 5$.

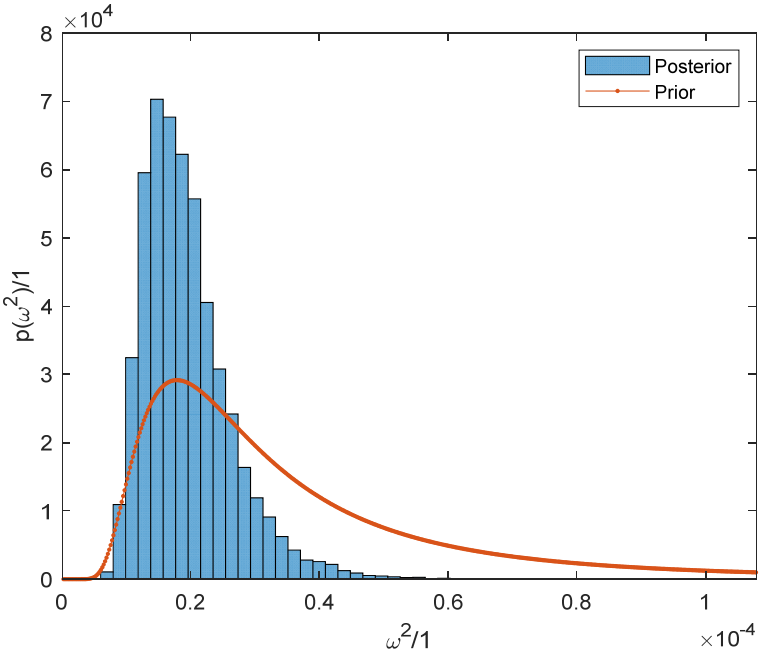

Figure 27: As Figure 24, except the prior distributions are defined by $m_0 = 5$.

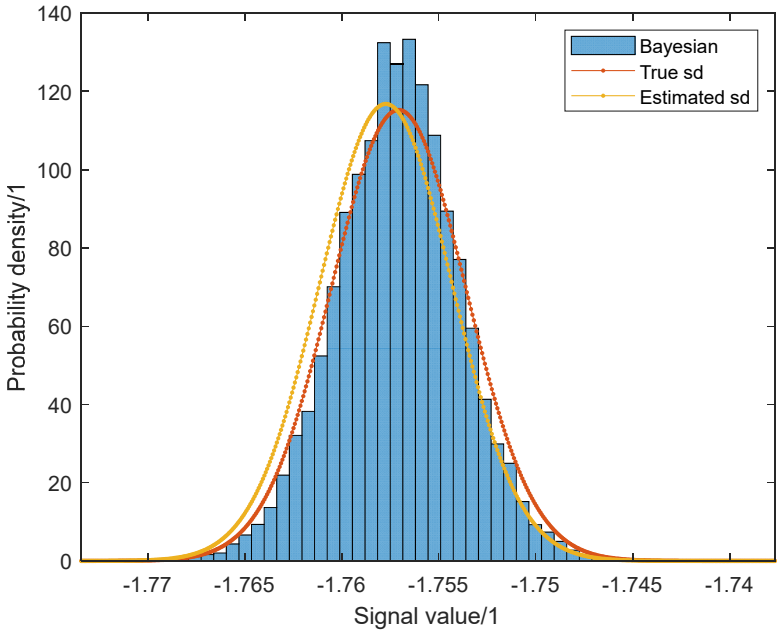

Figure 28: As Figure 25, except the prior distributions are defined by $m_0 = 5$.

7 DISCUSSION OF THE STRATH TESTBED

In practice, hardly any real-world development can go smoothly without a problem and always there is room for future extension. In this section, the few concerns about the STRATH testbed will be shared and a few observations will be discussed.

7.1 MODERATE DATASET CONCERNS

Due to the training required, the availability of a large dataset is generally desired for ML model development. However, in manufacturing, often obtaining a large training dataset implies higher overhead, which is the case for the STRATH testbed. To facilitate the development goal, instead of letting the forging machine run normally, as in data batch number 1, experiments have been designed by varying three controlled variables for a wider range of measurands. For the results, the difference brought by including data batch number 2 and data batch number 3 had a noticeable positive impact. In Figure 29, the signals of 'Force' for the all the three data batches are plotted superimposed on each other. In comparison with Figure 19, variations in the sensor signals became visible from this design of experiments (DoE).
Technology advances also provide possible solutions, such as transfer learning. That is, instead of learning from ground truth, learning starts from other artificial or even irrelevant data in a pre-training stage. But for large sensor networks with high accuracy requirements, the flexibility of this approach remains for further investigation. When ground truth is of such a limited size, the large-scale construction of synthetic time series, while it is interesting, needs careful consideration. It is left for future study.

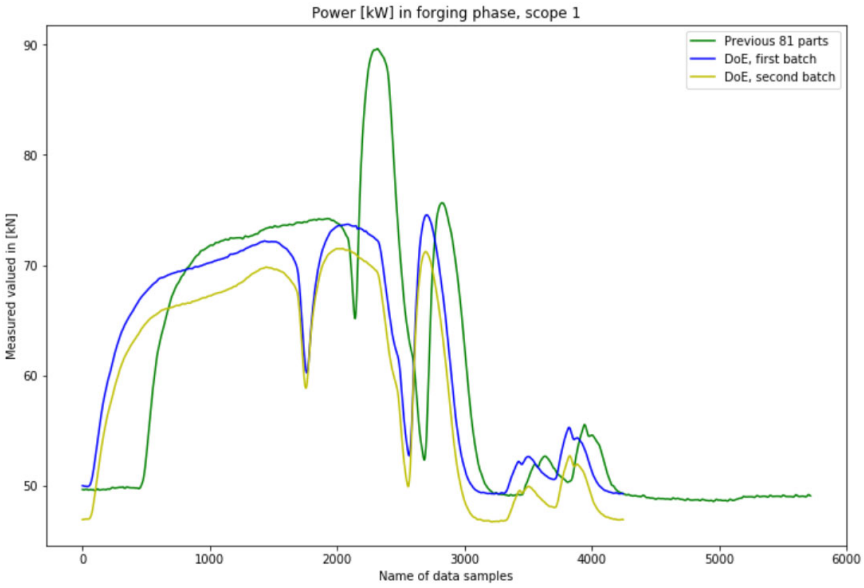

Figure 29: 'Force' signals for the three data batches.

7.2 REGRESSION, CLASSIFICATION, ACCURACY CONCERNS

ML is a well-known capability for classification. Therefore, assigning the measurands correct labels was one of the early tasks in the Met4FoF project when ML methods were considered. But studies showed that the grouping of the CMM dimensions was challenging. In Figure 30 and Figure 31, two CMM dimensions are plotted. Even with the DoE data, labelling the measurands was difficult. It is fair to judge that, even if the data were correctly grouped into different classes, there would be insufficient data in each class for training under high accuracy requirement.

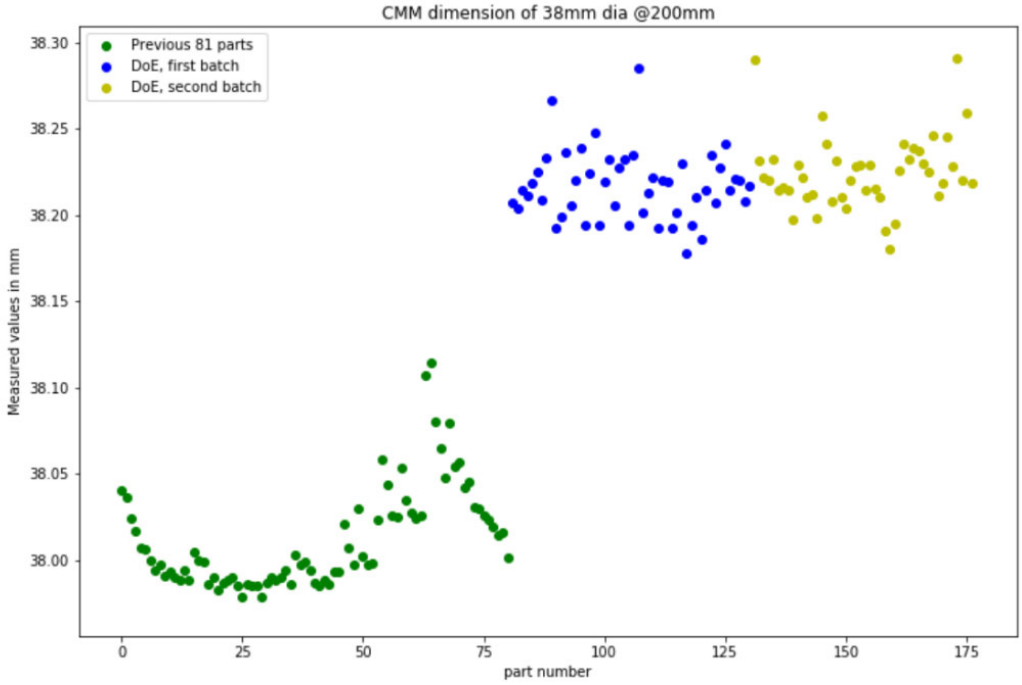

Figure 30: Examples of CMM dimensional data of "38mm dia@200mm"

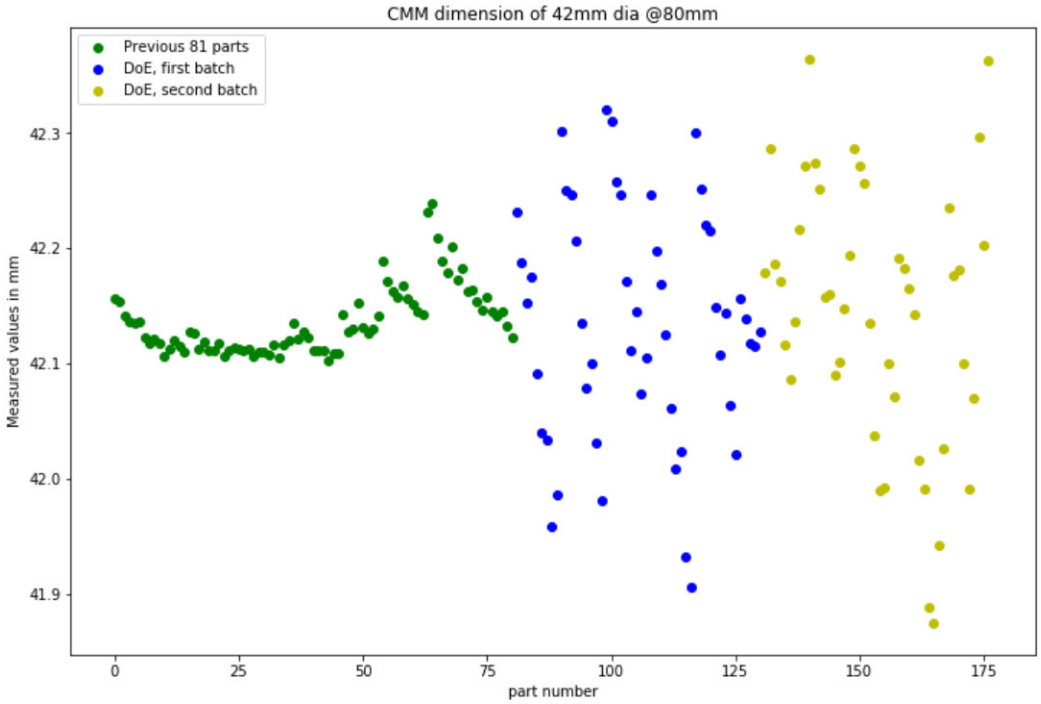

Figure 31: Examples of CMM dimensional data of "42mm dia@80mm".

This is substantially different from anomaly detection in manufacturing, where faulty data often stands out. Finding distinguishing features is expected to be more challenging in comparison. In fact, in data batch number 1, there was a faulty part which was excluded from the dataset in the end. Identifying the faulty part was a much more straightforward task to accomplish.

Some problems which could not be solved by classification may be treated by regression alternatively, as demonstrated in the STRATH testbed. ML approaches help in selecting the relevant features. Therefore, depending on the application, while advanced techniques such as deep learning providing a feasible alternative in some cases, conventional techniques such as regression may still play a role in the development phase.

7.3 PROCESS OPTIMIZATION

Besides the goal of CMM dimension prediction, it is also possible to study other process outputs for purposes of optimization. Energy consumption is one of those outputs. For the i^{th} part, it is computed as the integration of the signals measured by the 'Power' sensor $p(t)$ over the forging time T_i.

$$E_i = \int_0^{T_i} p_i(t)dt$$

For the three data batches from the STRATH testbed, energy consumption is shown in Figure 32. Due to the introduced variation of the controlled variables, energy consumption for the DoE has a wider spread.

Using the features described in section 5, preliminary results show that well used algorithms, such as Random Forest Regression, provide good prediction results with R^2 over 0.9, as shown in Figure 33. Additional investigation into how to utilize the prediction result for process optimization is left for further study.

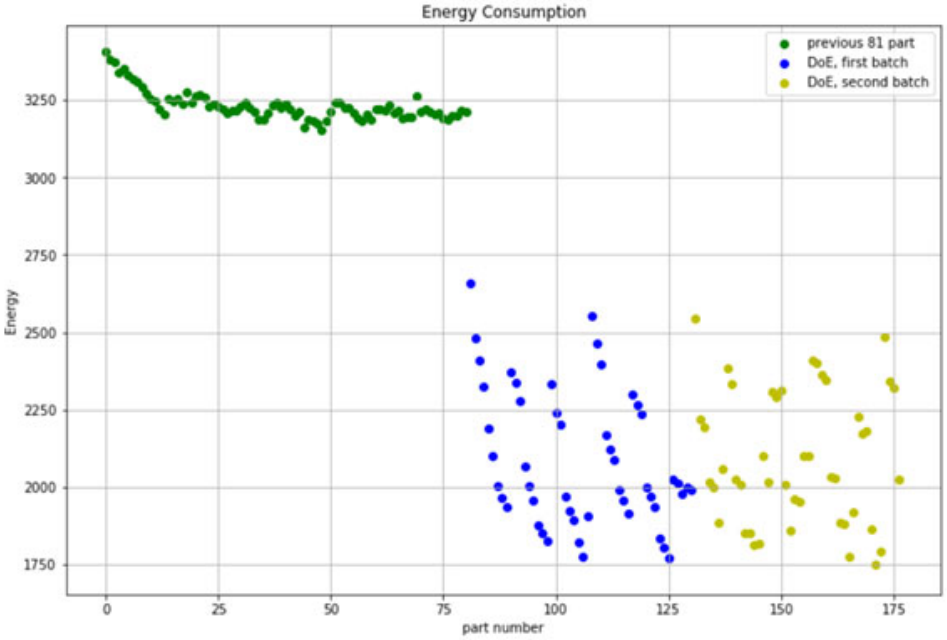

Figure 32: Energy consumption for the STRATH testbed.

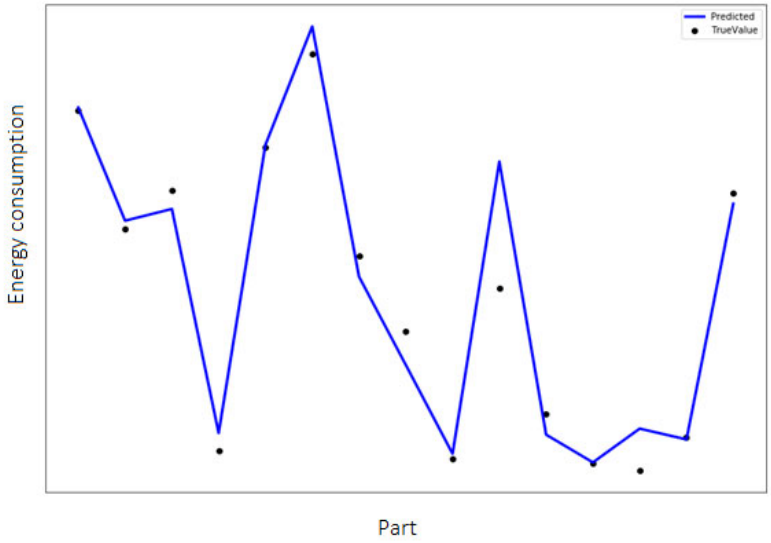

Figure 33: Simulation run on prediction of energy consumption for out-of-sample data.

8 SUMMARY

In this guide, the measurement coverage and accuracy required for process output quality targets in industrial sensor networks has been studied. The focus has been on the application of the Factory of the Future, in which automation, predictive maintenance and high accuracy production are expected. Although it is not possible to give a standard data processing chain, as it may vary for different applications, a few common processing stages have been described and studied with metrological uncertainty awareness in mind. These common processing stages include timing and synchronization, redundancy evaluation and feature selection, which are typical concerns for large sensor networks. For redundancy evaluation, the concepts of

measurement coverage and accuracy have been reviewed. It has been indicated how redundancy in an industrial sensor network can help to meet process output quality targets. An approach has been provided as an illustration of how to tackle timing and synchronization effects. Feature selection for mixed quality sensors has been studied. For sensors with varying degree of uncertainty, an example shows how the algorithm output can be interpreted.

The STRATH testbed, which is a radial forge at the University of Strathclyde's Advanced Forming Research Centre in the UK, has been used as an illustration when appropriate. Compared with the idealized user case of a large dataset for training and clear differentiation between classes for the measurand, the STRATH testbed was characterized by its modest dataset, large number of sensors and high accuracy requirement for the part dimensions. It may not be the best representative example of the timing and synchronization problem. However, alignment and segmentation are issues in defining the production phases. The modest dataset also presents a challenge to feature extraction and selection, and the utilization of more advanced approaches. But with a basic understanding of the forging process, many features have been extracted. A possible way for feature selection has been presented in this guide. Regression and cross validation have been carried out with the results presented. Those results demonstrated that even with a modest training dataset, good prediction became possible for a few CMM dimensions. A discussion of future development for the STRATH testbed has been included for the reader's interest. Finally, it cannot be stressed enough that the success of analysis methods can considerably depend on the problem studied, and therefore there may not be a single 'best' or 'always good' practice that works for every problem. There is always room for improvement.

ACKNOWLEDGEMENT

This project 17IND12 Met4FoF has received funding from the EMPIR programme co-financed by the Participating States and from the European Union's Horizon 2020 research and innovation programme.

REFERENCES

[Aha, 1996] Aha, D.W. and Bankert, R.L., “A comparative evaluation of sequential feature selection algorithms”, D. Fisher, J.-H. Lenz (Eds.), Artificial *Intelligence and Statistics* V, Springer, New York, 1996

[Breiman, 2001] Breiman, L., “Random forests”, Machine Learning, 45, pp. 5-32, 2001

[Carlos, 2018] Carlos, A. E., and Ruben, M.-M. “Machine learning techniques for quality control in high conformance manufacturing environment”, Advances in Mechanical Engineering, No. 10(2), 2018, DOI: 10.1177/1687814018755519

[Casella, 1985] Casella, G., “An introduction to empirical Bayes data analysis”, The American Statistician, 39(2), 83-87, 1985

[Chen, 2017] Chen, Y. “Novel Machine Learning Approaches for Modeling Variations in Semiconductor Manufacturing”, Thesis, Massachusetts Institute of Technology, 2017

[Chen, 2019] Chen, Y., and Boning, D., “Machine Learning Approaches for IC Manufacturing Yield Enhancement”, In book: Machine Learning in VLSI Computer-Aided Design, pp. 175-199, 2019, DOI: 10.1007/978-3-030-04666-8_6

[Columbus, 2020] Columbus, L. “10 ways AI is improving manufacturing in 2020”, Forbes, https://www.forbes.com/sites/louiscolumbus/?sh=1f6794b039e0, 2020

[Cox, 1993] Cox, M. G., Harris, P. M. and Humphreys, D. A., “An algorithm for the removal of noise and jitter in signals and its application to picosecond electrical measurement”, Numerical Algorithms, 5, pp. 491─508, 1993

[Cox, 2002] M. G. Cox, “The evaluation of key comparison data”, Metrologia, 39(6), 589, 2002

[Dorst, 2019] Dorst, T., Ludwig, B., Eichstädt, S., Schneider, T., and Schütze, A., "Metrology for the factory of the future: towards a case study in condition monitoring," IEEE International Instrumentation and Measurement Technology Conference (I2MTC), pp. 1-5, 2019, DOI: 10.1109/I2MTC.2019.8826973

[Dorst, 2021] Dorst, T., Robin, Y., Eichstädt, S., Schütze, A., and Schneider, T., "Influence of synchronization within a sensor network on machine learning results, " J. Sensors Sens. Syst., 10(2), pp. 233--245, 2021, DOI: 10.5194/jsss-10-233-2021.

[Eichstädt , 2021] Eichstädt, S., Gruber, M., Vedurmudi, A.P., Seeger, B., Bruns, T., and Kok, G. “Toward Smart Traceability for Digital Sensors and the Industrial Internet of Things”, Sensors 2021; 21(6):2019. DOI: 10.3390/s21062019

[Gelman, 1992] Gelman, A., and Rubin, D. B., “A single series from the Gibbs sampler provides a false sense of security”, Bayesian statistics, 4, pp. 625-631, 1992

[Geweke, 1992] Geweke, J., “Evaluating the accuracy of sampling-based approaches to the calculations of posterior moments”, Bayesian statistics, 4, pp. 641-649, 1992

[Gnana, 2016] Gnana, D. A. A., Balamurugan, S. A. A., and Leavline, E. J., “Literature review on feature selection methods for high-dimensional data”, International Journal of Computer Applications, 136(1), pp. 9-17, 2016

[Görtler, 2019] Görtler, J., Spinner, T., Streeb, D., Weiskopf, D., and Deussen, O., “Uncertainty-Aware Principal Component Analysis”, IEEE Transactions on Visualization and Computer Graphics, 26 (1), 822-831, 2019

[GUM, 2008] BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML. “Evaluation of measurement data — Guide to the expression of uncertainty in measurement”. Joint Committee for Guides in Metrology, JCGM, 2008

[Hall, 1999] Hall, M. A., “Correlation-based feature selection for machine learning”, 1999

[Haris 2019] Lulic, H., “Strathcylde AFRC machine learning tutorials”, 2019, in Github: https://github.com/harislulic/Strathcylde_AFRC_machine_learning_tutorials

[Helwig, 2017] Helwig, N., Schneider, T., and Schütze, A., “MoSeS-Pro: Modular sensor systems for real time process control and smart condition monitoring using XMR-technology,” Proc. 14th Symp. Magnetoresistive Sensors Magn. Syst., Wetzlar, Germany, March 21- 23, 2017

[Hill, 2020] Hill, J., Linero, A. and Murray, J., “Bayesian Additive Regression Trees: A review and look forword”, Annual Review of Statistics and Its Application, 7, pp. 251-278, 2020, DOI: 10.1146/annurev-statistics-031219-041110

[Hoffman, 2014] Hoffman, M. D., and Gelman, A., “The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo”, Journal. Mach. Learn. Res., 15(1), pp. 1593-1623, 2014

[Jović, 2015] Jović, A., Brkić, K., and Bogunović, N. “A review of feature selection methods with applications”, In IEEE 38th international convention on information and communication technology, electronics and microelectronics (MIPRO), pp. 1200-1205, 2015

[Kok, 2020a] Kok, G., and Harris, P., "Quantifying Metrological Redundancy in an Industry 4.0 Environment", IEEE International Workshop on Metrology for Industry 4.0 & IoT, pp. 464-468, 2020, DOI: 10.1109/MetroInd4.0IoT48571.2020.9138235

[Kok, 2020b] Kok, G., and Harris, P., "Uncertainty Evaluation for Metrologically Redundant Industrial Sensor Networks", IEEE International Workshop on Metrology for Industry 4.0 & IoT, pp. 84-88, DOI: 10.1109/MetroInd4.0IoT48571.2020.9138297, 2020

[Liu, 2016] Liu, Q., and Wang, D., “Stein variational gradient descent: A general purpose Bayesian inference algorithm”, 2016, arXiv preprint arXiv:1608.04471

[Moller, 2017] Moller, E. “The use of machine learning in industrial quality control”, Thesis, KTH Royal Institute of Technology, 2017

[Namuduri, 2020] Namuduri, S. Narayanan, B., Davuluru, V., Burton, L. and Bhansali, S., “Review ─ Deep learning methods for sensor based predictive maintenance and future perspectives for electrochemical sensors”, Journal of The Electrochemical Society, 167(3), 2020

[Robnik-Sikonja, 2003] Robnik-Sikonja, M., and Kononenko, I., “Theoretical and empirical analysis of ReliefF and RReliefF”, Machine Learning, 53, pp. 23–69, 2003

[Schneider, 2018A] Schneider, T., Helwig, N., and Schütze, A., “Industrial condition monitoring with smart sensors using automated feature extraction and selection,” Meas. Sci. Technol., 29(9), 2018

[Schneider, 2018B] Schneider, T., Helwig, N., Klein, S., and Schütze, A., “Influence of sensor network sampling rate on multivariate statistical condition monitoring of industrial machines and processes,” pp. 2–5, 2018, DOI:10.3390/proceedings2130781

[Shrouf, 2014], Shrouf, F., Ordieres, J. and Miragliotta, G., "Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm," IEEE International Conference on Industrial Engineering and Engineering Management, Selangor, Malaysia, 2014, pp. 697-701, 2014, DOI: 10.1109/IEEM.2014.7058728

[Subrahmanya, 2010] Subrahmanya, N., Shin, Y. C., and Meckl, P. H., “A Bayesian machine learning method for sensor selection and fusion with application to on-board fault diagnostics”, Mechanical systems and signal processing, 24(1), pp. 182-192, 2010

[Tachtatzis , 2019] Tachtatzis, C., Gourlay, G., Andonovic, I., and Panni, O., “Sensor data set radial forging at AFRC testbed v2”, 2019, DOI: 10.5281/zenodo.3405265

[Tipping, 2001] Tipping, M., “Sparse Bayesian Learning and the Relevance Vector Machine”, *Journal of Machine Learning Research*, 1, pp. 211-244, 2001

[Venkatesh, 2019] Venkatesh, B., and Anuradha, J., “A review of feature selection and its methods”, Cybernetics and Information Technologies, 19(1), pp. 3-26, 2019

[White, 2017] White, D.R., “Propagation of Uncertainty and Comparison of Interpolation Schemes”, International Journal Thermophys pp. 38-39, 2017, DOI: 10.1007/s10765-016-2174-6

[Wold, 1987] Wold, S., Esbensen, K., and Geladi, P., “Principal Component Analysis”, Chemometrics and Intelligent Laboratory Systems, 2, pp. 37-52, 1987

[Wuest, 2014] Wuest, T., Irgens, C. and Thoben, K.-D., “An approach to quality monitoring in manufacturing using supervised machine learning on product state data”, Journal of Intelligent Manufacturing, 25, pp. 1167-1180, 2014

[Wuest, 2016] Wuest, T., Weimer, D., Irgens, C. and Thoben, K.-D., “Machine learning in manufacturing: advantages, challenges, and applications”, Production & Manufacturing Research, 4(1), pp. 23-45, 2016, DOI: 10.1080/21693277.2016.1192517